

NOTE

The following list of changes were made as a result of errors found in this publication. Replace the pages as indicated below.

<u>PAGE</u>	<u>DESCRIPTION</u>
i - xiv	Table of contents updated.
14-3	Reference to MODBUS network removed.
14-6	Reference to Allen-Bradley removed.
Ch. 23	Chapter reserved for future release.
Ch. 24	4800-E18 Mitsubishi direct connect added.

Publication Change Notice

DATE: March 19, 1991

PCN NUMBER: 140

PUBLICATION NUMBER: 90444-999

TITLE: 4800-Ex Interface Modules

PREVIOUS PCNs BY NUMBER: 132

EFFECTIVE DATE: immediately

This Publication Change Notice provides updates for the publication specified above. These changes will remain in effect for subsequent releases unless specifically amended by another PCN or superseded by a publication revision.

NOTE:

Please file this page at the back of your manual to provide a record of the changes. Insert all change pages into the manual and dispose of the obsolete pages.

Publication Change Notice

NOTE

The following changes were made to document the newly released OMRON PLC direct connect. Add all the pages of this PCN to your manual, and discard the old pages as directed. The new or replacement pages have a delta (Δ) in the footer to distinguish them from the original pages. The table below indicates what information changed.

<u>PAGES</u>	<u>DESCRIPTION</u>
i-xvi	Updated Table of Contents. Discard original Table of Contents and ignore Table of Contents in PCN #132; insert entire Table of Contents, dated March, 1991, from this PCN.
23-1 to 23-13	Documentation for OMRON direct connect. Insert entire chapter behind Chapter 22.

NOTE

Please make sure your Table of Contents is the most current, dated March, 1991 (from PCN #140).

Publication Change Notice

DATE: July 8, 1991

PCN NUMBER: 149

PUBLICATION NUMBER: 90444-999

TITLE: 4800-Ex Interface Modules

PREVIOUS PCNs BY NUMBER: 132, 140

EFFECTIVE DATE: Immediately

This Publication Change Notice provides updates for the publication specified above. These changes will remain in effect for subsequent releases unless specifically amended by another PCN or superseded by a publication revision.

NOTE:

Please file this page at the back of your manual to provide a record of the changes. Insert all change pages into the manual and dispose of the obsolete pages.

Publication Change Notice

NOTE

The following changes were made to update the 4800-E9 chapter and document the new 4800-E19 TI 305 and 4800-E21 GEM 80 direct connects. Add all the pages of this PCN to your manual, and discard the old pages as directed. The new or replacement pages have a delta (Δ) in the footer to distinguish them from the original pages. The table below indicates which pages have changed.

<u>PAGES</u>	<u>DESCRIPTION</u>
i-xvii	Updated Table of Contents. Discard all other Table of Contents pages, and insert entire Table of Contents, dated July, 1991, from this PCN.
Chapter 15	Updated information for Eagle Signal PLC direct connect.
Chapter 25	New chapter for Texas Instruments Series 305 PLC direct connect.
Chapter 26	Chapter reserved for GE Series 90 PLC.
Chapter 27	New chapter for GEM 80 PLC direct connect.

NOTE

Please make sure Table of Contents is the most current, dated July, 1991 (from PCN #149).

XYCOM REVISION RECORD

<i>Revision</i>	<i>Description</i>	<i>Date</i>
A	Manual Released	1/90

Trademark Information

- *IBM PC, AT, and XT are registered trademarks of the International Business Machines Corporation*

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 does not guarantee the accuracy of the information and makes no commitment toward keeping it up to date.

Address comments concerning
this manual to:

xycom

Technical Publications Dept.
750 North Maple Road
Saline, Michigan 48176

*Part Number: **90444-999 A***

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
1	4800-Ex EXPANSION MODULE	
1.1	Introduction	1-1
1.2	4800-Ex Expansion Module Overview	1-2
1.3	Specifications	1-4
2	4800-Ex EXPANSION MODULE INSTALLATION	
2.1	Introduction	2-1
2.2	Installation	2-2
2.3	Communication Module Installation	2-8
2.4	Verifying Installation	2-11
2.5	Power-Up or Reset	2-12
2.6	Ports	2-12
2.7	Connecting the Optional Keyboard	2-12
3	BASIC CONCEPTS	
3.1	Introduction	3-1
3.2	Menu Hierarchy	3-1
3.3	Main Menu	3-3
3.4	Configuration Menu	3-4
3.4.1	Serial Port Configuration Menu (1st Configuration Menu)	3-4
3.4.2	Miscellaneous Configuration Menu (3rd configuration Menu)	3-7
3.5	Program Utilities Menu	3-10
3.5.1	Using the Editor	3-10
3.5.2	Program Execution	3-13
3.5.3	Copying Program Blocks	3-14
3.5.4	Backup/Restore Programs	3-14
3.5.5	Using the Directory	3-15
3.5.6	Erase	3-16
3.5.7	Search	3-16
3.6	Backup/Restore Menu	3-16
3.7	Diagnostics Menu	3-17
3.7.1	RAM Test	3-17
3.7.2	ROM Checksum	3-18
3.7.3	Character Attributes Test	3-19
3.7.4	Serial Loop Back Test	3-19

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
3.7.5	CRT Alignment Pattern	3-19
3.7.6	CRT Brightness Pattern	3-19
3.7.7	Clock/Calendar Test	3-20
3.7.8	Continuous Test	3-20
3.8	Terminal Mode Menu	3-21
3.9	Set Password Menu	3-21
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)	4-8
4.4	Transferring Data To/From the Terminal	4-10
4.4.1	Data Registers	4-11
4.4.2	Transferring Data to the Terminal	4-11
z 4.4.3	Requesting Data from the Terminal	4-12
4.4.4	Transmitting Data from the Terminal	4-12
4.5	Programming Rules	4-14
4.5.1	Communication Status Register	4-15
4.6	Commands	4-16
4.6.1	User Interface Considerations	4-16
4.6.2	Screen Text ("...")	4-16
4.6.3	Display Register Value (# or \$)	4-17
4.6.4	Position Cursor (@)	4-18
4.6.5	Label(%)	4-19
4.6.6	Add to Register (ADD)	4-20
4.6.7	AND	4-21
4.6.8	Beep (BP)	4-22
4.6.9	Draw Box (BX)	4-23
4.6.10	Clear Buffer (CB)	4-24
4.6.11	Clear Line (CL)	4-25
4.6.12	Clear Bit (CLRB)	4-26
4.6.13	Clear Screen (CS)	4-27
4.6.14	Clear Window (CW)	4-28
4.6.15	Down (D)	4-29
4.6.16	Decrement Data Register (DEC)	4-30
4.6.17	Divide Register (DIV)	4-31
4.6.18	Execute Sub-program Block (ES)	4-32
4.6.19	Execute Program Block (EX)	4-33
4.6.20	Exit from Program Block (Exit)	4-34
4.6.21	Draw Filled Box (FBX)	4-35

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
4.6.22	Go (GO)	4-36
4.6.23	Horizontal Bar Left (HBL)	4-37
4.6.24	Horizontal Bar Right (HBR)	4-38
4.6.25	Draw Horizontal Line (HL)	4-39
4.6.26	Conditional (IF)	4-40
4.6.27	IF Bit (IFB)	4-42
4.6.28	Increment Data Register (INC)	4-43
4.6.29	KEXIT	4-44
4.6.30	Keypad Status (KS)	4-45
4.6.31	Left (L)	4-46
4.6.32	Remainder (MOD)	4-47
4.6.33	Multiply Register (MUL)	4-48
4.6.34	New Line (NL)	4-49
4.6.35	NOT	4-50
4.6.36	ONKEY	4-51
4.6.37	OR	4-52
4.6.38	Plot (Plot)	4-53
4.6.39	Pause (PS)	4-54
4.6.40	Right (R)	4-55
4.6.41	Reset Attributes (RE)	4-56
4.6.42	Restore Attributes (RSTA)	4-57
4.6.43	Save Attributes (SAVA)	4-58
4.6.44	Set Background Color (SBC)	4-59
4.6.45	Set Attributes (SE)	4-60
4.6.46	Set Bit (SETB)	4-63
4.6.47	Set Foreground Color (SFC)	4-64
4.6.48	Stop Program Execution (STOP)	4-65
4.6.49	Subtract from Register (SUB)	4-66
4.6.50	Transfer Data (TR)	4-67
4.6.50.1	Pictures	4-68
4.6.50.2	Data Types	4-69
4.6.50.3	Transferring Text Characters	4-70
4.6.50.4	Numeric Data Types	4-73
4.6.51	Unplot	4-82
4.6.52	Up (U)	4-83
4.6.53	Vertical Bar Up (VBU or VB)	4-84
4.6.54	Vertical Bar Down (VBD)	4-85
4.6.55	Draw Vertical Line (VL)	4-86
4.6.56	XOR	4-87
4.7	Sample Program	4-88

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
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	5-13
5.7	Keyboard Translation	5-13
5.8	Performance Hints	5-13
6	REMOTE COMMANDS	
6.1	Introduction	6-1
6.2	Serial Remote Commands	6-2
6.2.1	Reset	6-3
6.2.2	Execute Program Block	6-4
6.2.3	Execute Subprogram Block	6-5
6.2.4	Transfer from Data Register	6-6
6.2.5	Transfer to Data Register	6-7
6.2.6	Load Time and Date	6-8
6.2.7	Load All Registers	6-9
6.2.8	Send Password	6-10
6.2.9	Receive Program Block (Serial Port only)	6-11
6.2.10	Send a Number of Registers	6-12
6.2.11	Receive a number of Registers	6-13
6.3	How Remote Commands are Processed	6-14
7	4800-E1: SERIAL EXPANSION MODULE	
7.1	Introduction	7-1
7.2	Configuration Menu	7-1
7.3	Serial Loop Back Test	7-3

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
7.4	Printer Port Commands	7-4
7.4.1	Enable Printer Port	7-4
7.4.2	Disable Printer Port	7-4
7.4.3	Enable Screen	7-5
7.4.4	Disable Screen	7-5
7.5	Remote Commands	7-6
7.5.1	Execute Stored Screen	7-6
7.5.2	Receive Stored Screen	7-6
7.5.3	Transmit Stored Screen	7-7
7.5.4	Jump to Stored Screen	7-7
7.5.5	Copy Screen Program	7-7
7.5.6	Set Programmable Key	7-8
7.5.7	Set Executable Key	7-9
7.5.8	Enable Output Echo	7-9
7.5.9	Enable Output Echo	7-11
7.6	Stored Screen Utilities Menu	7-12
7.7	Screen Editor	7-13
7.8	Nested Screens	7-16
 4800-E2: MULTI-DROP EXPANSION MODULE		
8.1	Introduction	8-1
8.2	Configuration Menu	8-1
8.3	Serial Loop Back Test	8-4
8.4	Multidrop Network Overview	8-7
8.4.1	Multidrop Commands	8-8
8.4.2	Operator Input	8-8
8.4.3	Keyboard Echo and Keyboard Character Mode	8-8
8.4.4	Terminal Addressing and Broadcasting	8-8
8.4.5	Multidrop System Consideration	8-9
8.4.6	Using Remote Commands With a Multidrop Network	8-10
8.5	Connecting a Multidrop Network	8-11
8.5.1	Head-end Termination	8-12
8.5.2	Tail-end Termination	8-12
8.5.3	RS-485 Cabling	8-12
8.6	Multidrop Commands	8-13

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
8.6.1	Select for Input	8-13
8.6.2	Select for Output	8-13
8.6.3	Enable Keyboard Echo	8-14
8.6.4	Disable Keyboard Echo	8-14
8.6.5	Enable Keyboard Character Mode	8-15
8.6.6	Disable Keyboard Character Mode	8-15
8.7	Stored Screen Utilities Menu	8-16
8.8	Screen Editor	8-17
8.9	Nested Screens	8-20
9	4800-E3: PARALLEL EXPANSION MODULE	
9.1	Introduction	9-1
9.2	Configuration Menu	9-1
9.3	Communications Status Register	9-1
9.4	Cabling the Parallel Port	9-2
9.5	Special Commands	9-6
9.6	Data Transfer Between a Programmable Controller and the Parallel Port	9-6
9.6.1	Parallel Input Port	9-6
9.6.2	Parallel Output Port	9-8
9.7	Parallel Remote Commands	9-10
9.7.1	Reset	9-11
9.7.2	Execute Program	9-11
9.7.3	Transfer from Data Register	9-11
9.7.4	Transfer to Data Register	9-11
9.7.5	Short Transfer from Data Register	9-12
9.7.6	Short Transfer to Data Register	9-12
10	4800-E4: SERIAL EXPANSION MODULE	
10.1	Introduction	10-1
10.2	Configuration Menu	10-1
10.3	Communications Status Register	10-1
10.4	Cabling the Serial Ports	10-2
10.5	Special Commands	10-3

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
11	4800-E5: MULTIDROP EXPANSION MODULE	
11.1	Introduction	11-1
11.2	Configuration Menu	11-2
11.3	Serial Loop Back Test	11-3
11.4	Multidrop Network Overview	11-6
11.4.1	Multidrop Commands	11-7
11.4.2	Operator Input	11-7
11.4.3	Keyboard Echo and Keyboard Character Mode	11-7
11.4.4	Terminal Addressing and Broadcasting	11-8
11.4.5	Multidrop System Consideration	11-8
11.4.6	Using Remote Commands With a Multidrop Network	11-9
11.5	Connecting a Multidrop Network	11-10
11.5.1	Head-end Termination	11-11
11.5.2	Tail-end Termination	11-11
11.5.3	RS-485 Cabling	11-11
11.6	Multidrop Commands	11-12
11.6.1	Select for Input	11-12
11.6.2	Select for Output	11-12
12	4800-E6: ALLEN-BRADLEY DATA HIGHWAY	
12.1	Introduction	12-1
12.2	Cabling the Terminal to the PLC	12-2
12.3	Configuration Menu	12-4
12.4	Serial Loop Back Test	12-6
12.5	Communication Status Register	12-7
12.6	Commands	12-8
12.6.1	Get Data From Data Table (GET)	12-8
12.6.2	Get Diagnostic Status (GETDIAG)	12-9
12.6.3	Get a Bit of Data From Data Table (GETIO)	12-10
12.6.4	Write Data to a Data Table (PUT)	12-11
12.6.5	Get a Bit of Data to an Address (PUTIO)	12-12

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
13	4800-E7: MODICON MODBUS NETWORK	
13.1	Introduction	13-1
13.2	Cabling the Terminal to the PLC	13-2
13.3	Configuration Menu	13-4
13.4	Serial Loop Back Test	13-5
13.5	Communication Status Register	13-6
13.6	Commands	13-7
13.6.1	Get Input Register (GETIR)	13-7
13.6.2	Get Input Status (GETIS)	13-8
13.6.3	Get Output Register (GETOR)	13-9
13.6.4	Get Input Status (GETOS)	13-10
13.6.5	Modify Multiple Registers (PUTRM)	13-11
13.6.6	Modify a Single Register (PUTRS)	13-12
14	4800-E8: TEXAS INSTRUMENTS SERIES 500	
14.1	Introduction	14-1
14.2	Cabling the Terminal to the PLC	14-2
14.3	Configuration Menu	14-3
14.4	Serial Loop Back Test	14-4
14.5	Communication Status Register	14-5
14.6	Commands	14-6
14.6.1	Get PLC Data (GET)	14-6
14.6.2	Get PLC I/O Point (GETIO)	14-7
14.6.3	Get PLC Register Block (GETR)	14-8
14.6.4	Put PLC Data (PUT)	14-9
14.6.5	Put PLC I/O Point (PUTIO)	14-10
14.6.6	Put PLC Register Block (PUTR)	14-11
15	4800-E9: EAGLE SIGNAL PLC NETWORK	
15.1	Introduction	15-1
15.2	Cabling the Terminal to the PLC	15-2
15.3	Configuration Menu	15-3
15.4	Serial Loop Back Test	15-4
15.5	Communication Status Register	15-5

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
15.6	Commands	15-6
15.6.1	Get PLC Data (GET)	15-6
15.6.2	Get I/O Point (GETIO)	15-9
15.6.3	Get Register Block (GETR)	15-11
15.6.4	Put Data (PUT)	15-12
15.6.5	Put Register Block (PUTR)	15-13
16	4800-E10: SQUARE-D SY/MAX NETWORK	
16.1	Introduction	16-1
16.2	Cabling the Terminal to the PLC	16-2
16.3	Configuration Menu	16-4
16.4	Serial Loop Back Test	16-5
16.5	Communication Status Register	16-6
16.6	Commands	16-7
16.6.1	Get Register from Destination (GET)	16-8
16.6.2	Get Register from Destination Device (GETN)	16-9
16.6.3	Get Register Using Net to Net (GETNN)	16-9
16.6.4	Write to a Data Register (PUT)	16-10
16.6.5	Write a Register from the Destination Device (PUTN)	16-11
16.6.6	Write a Register Using Net to Net (PUTNN)	16-12
17	4800-E11: WESTINGHOUSE NUMA-LOGIC NETWORK	
17.1	Introduction	17-1
17.2	Cabling the Terminal to the PLC	17-2
17.3	Configuration Menu	17-3
17.4	Serial Loop Back Test	17-4
17.5	Communication Status Register	17-5
17.6	Commands	17-6
17.6.1	Get PLC Data (GET)	17-6
17.6.2	Put PLC Data (PUT)	17-7
17.6.3	Put PLC I/O Point (PUTIO)	17-8

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
18	4800-E12: GENERAL ELECTRIC PLC NETWORK	
18.1	Introduction	18-1
18.2	Cabling the Terminal to the PLC	18-2
18.3	Configuration Menu	18-6
18.4	Serial Loop Back Test	18-7
18.5	Communication Status Register	18-8
18.6	Commands	18-9
18.6.1	Get Register (GETR)	18-9
18.6.2	Get Input Status (GETIS)	18-10
18.6.3	Get Output Status (GETOS)	18-11
18.6.4	Put Registers (PUTR)	18-12
18.6.5	Put Output Status (PUTOS)	18-13
19	4800-E13: SIEMENS PLC NETWORK	
19.1	Introduction	19-1
19.2	Cabling the Terminal to the PLC	19-2
19.3	Configuration Menu	19-3
19.4	Serial Loop Back Test	19-4
19.5	Communication Status Register	19-5
19.6	Commands	19-6
19.6.1	Get PLC Data (GET)	19-7
19.6.2	Get Data Block (GETDB)	19-8
19.6.3	Get Timer/Counter Data (GETTC)	19-9
19.6.4	Put Data Block (PUTDB)	19-10
19.6.5	Compress Data Block (COMDB)	19-11
20	4800-E14: ALLEN-BRADLEY T3 PORT	
20.1	Introduction	20-1
20.2	Cabling the Terminal to the PLC	20-2
20.3	Configuration Menu	20-3
20.4	Serial Loop Back Test	20-4
20.5	Communication Status Register	20-5
20.6	Commands	20-6
20.6.1	Get Data From Data Table (GET)	20-6
20.6.2	Get a Bit of Data From Data Table (GETIO)	20-7
20.6.3	Get Diagnostic Status (GETDIAG)	20-8
20.6.4	Write Data to a Data Table (PUT)	20-9
20.6.5	Get a Bit of Data to an Address (PUTIO)	20-10

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
21	4800-E15: CINCINNATI MILICRON	
21.1	Introduction	21-1
22	4800-E16: TEXAS INSTRUMENTS SERIES 405 PLC	
22.1	Introduction	22-1
22.2	Cabling the Terminal to the PLC	22-2
22.3	Configuration Menu	22-3
22.4	Serial Loop Back Test	22-4
22.5	Communication Status Register	22-5
22.6	Commands	22-6
22.6.1	Get Data From Data Table (GET)	22-7
22.6.2	Write Data to a Data Table (PUT)	22-8
23	4800-E17: OMRON	
23.1	Introduction	23-1
23.2	Cabling the Terminal to the PLC	23-2
23.3	Configuration Menu	23-4
23.4	Communication Status Register	23-5
23.5	Commands	23-7
23.5.1	Read and Clear Host Link Unit Error Conditions (GETER)	23-8
23.5.2	Abort Current Operations and Re-Initialize (INIT)	23-11
23.5.3	Read from PLC Register Areas (GET)	23-12
23.5.4	Write to PLC Register Areas (PUT)	23-13
23.5.5	Change PLC Mode	23-14

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
24	4800-E18: MITSUBISHI MELSEC-A	
24.1	Introduction	24-1
24.2	Cabling the Terminal to the PLC	24-2
24.3	Configuration Menu	24-3
24.4	Communication Status Register	24-4
24.5	Commands	24-5
24.5.1	Get PLC Device Bits (GETDB)	24-5
24.5.2	Get PLC Device Word (GETDW)	24-6
24.5.3	Get Buffer Memory Values (GETBUF)	24-7
24.5.4	Get PC CPU Type (GETCPU)	24-8
24.5.5	Get Parameter Memory (GETPAR)	24-9
24.5.6	Get Special Function Unit Buffer Memory (SETSFU)	24-10
24.5.7	Put PLC Device Bits (PUTDB)	24-11
24.5.8	Put PLC Device Word (PUTDW)	24-12
24.5.9	Put Buffer Memory Values (PUTBUF)	24-13
24.5.10	Put PC Run/Stop Command (PUTCPU)	24-14
24.5.11	Put Parameter Memory (PUTPAR)	24-15
24.5.12	Put Special Function Unit Buffer Memory (PUTSFU)	24-16
24.6	Program Example	24-17
25	4800-E19: TEXAS INSTRUMENTS SERIES 305	
25.1	Introduction	25-1
25.2	Cabling the Terminal to the PLC	25-2
25.3	Configuration Menu	25-3
25.4	Serial Loop Back Test	25-4
25.5	Communication Status Register	25-5
25.6	Commands	25-6
25.6.1	Get Register (GETR)	25-6
25.6.2	Get I/O Status (GETIO)	25-7
25.6.3	Put Register (PUTR)	25-8
25.6.4	Put I/O Status (PUTIO)	25-9
26	4800-E20: GE SERIES 90 PLC	
26.1	Introduction	

TABLE OF CONTENTS (continued)

CHAPTER	TITLE	PAGE
27		
27.1	Introduction	27-1
27.2	Cabling the Terminal to the PLC	27-2
27.3	Configuration Menu	27-3
27.4	Serial Loop Back Test	27-4
27.5	Communication Status Register	27-5
27.6	Commands	27-6
27.6.1	Read a Word (GETWD)	27-6
27.6.2	Read a Block (GETBLK)	27-7
27.6.3	Write a Word (PUTWD)	27-8
27.6.4	Write a Block (PUTBLK)	27-8
27.6.5	Exchange Blocks (SWAPJK)	27-9

APPENDICES

A	SAVING/LOADING/VERIFYING SCREEN PROGRAMS
B	KEYPAD KEY CODES
C	OIL COMMANDS

LIST OF FIGURES

FIGURE	TITLE	PAGE
2-1	Removing the Terminal Cover	2-2
2-2	Overhead View of Terminal Components	2-3
2-3	Rear Cover Plate Removal	2-4
2-4	Battery Backup Activation	2-5
2-5	Installation of the Expansion Module	2-6
2-6	Installation of Personality Plates	2-7
2-7	Rear Cover Plate Removal	2-8
2-8	Communication Adapter Installation	2-9
2-9	Communication Adapter Connectors	2-10
2-10	Location of Switch S1	2-13
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	3-7
3-5	Program Utilities Menu	3-10
3-6	Cursor Movement Characters	3-11
3-7	Diagnostics Menu	3-17
4-1	The Block Command Transfer (TR)	4-12
4-3	Screen Created by Sample Program	4-89
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	5-11
7-1	Main Menu	7-1
7-2	Configuration Menu	7-2
7-3	Serial Port Test Plugs	7-3
7-4	Nested Screen Example	7-16
8-1	Main Menu	8-1
8-2	Configuration Menu	8-3
8-3	Serial Port Test Plugs	8-4
8-4	Test Cable 1	8-5
8-5	Test Cable 2	8-6
8-6	Multidrop Connection	8-11
8-7	Nested Screen Example	8-20
9-1	Configuration Menu	9-1
9-2	Parallel I/O Port Pinouts	9-2
9-3	Typical Wiring: Terminal to PLC	9-5

LIST OF FIGURES (continued)

FIGURE	TITLE	PAGE
11-1	Configuration Menu	11-2
11-2	Serial Port Test Plugs	11-3
11-3	Test Cable 1	11-4
11-4	Test Cable 2	11-5
11-5	Multidrop Connection	11-10
12-1	Connections	12-2
12-2	Configurations	12-3
12-3	Configuration Menu	12-4
12-4	Serial Port Test Plugs	12-6
13-1	Connections	13-2
13-2	Configuration Menu	13-3
13-3	Serial Port Test Plugs	13-4
14-1	Connections	14-2
14-2	Configuration Menu	14-3
14-3	Serial Port Test Plugs	14-4
15-1	Connections	15-2
15-2	Configuration Menu	15-3
15-3	Serial Port Test Plugs	15-4
16-1	Connections	16-2
16-2	Configuration Menu	16-3
16-3	Serial Port Test Plugs	16-4
17-1	Connections	17-2
17-2	Configuration Menu	17-3
17-3	Serial Port Test Plugs	17-4
18-1	GE Series One and Three PLCs	18-2
18-2	GE Series Five PLC, RS-232 Programming Port	18-3
18-3	GE Series Five PLC, RS-422 Port J1	18-4
18-4	GE Series Five PLC, RS-422 Port J2	18-4
18-5	GE Series Six PLC, RS-232 Programming Port	18-3
18-6	GE Series Six PLC, RS-422 Port J1	18-4
18-7	GE Series Six PLC, RS-422 Port J2	18-4
18-8	Configuration Menu	18-5
18-9	Serial Port Test Plugs	18-6
19-1	Connections	19-2
19-3	Serial Port Test Plugs	19-4

LIST OF FIGURES (continued)

FIGURE	TITLE	PAGE
20-1	Connections	20-2
20-3	Serial Port Test Plugs	20-4
22-1	Connections	22-2
22-3	Serial Port Test Plugs	22-4
23-1	RS-232C Connection	23-2
23-2	RS-422 Connection	23-3
23-3	Configuration Menu	23-4
24-1	Connections	24-2
24-2	Configuration Menu	24-3
25-1	Connections	25-2
25-2	Configuration Menu	25-3
25-3	Serial Port Test Plugs	25-4
27-1	Connections	27-2
27-2	Configuration Menu	27-3
27-3	Serial Port Test Plugs	27-4

LIST OF TABLES

TABLE	TITLE	PAGE
4-1	OIL Commands	4-1
4-2	OIL Commands by Function	4-3
5-1	Registers #8 and #9	5-7
6-1	Serial Remote Commands	6-2
7-1	Keypad Key Numbering	7-8
8-1	RS-485 Transmission Lines	8-12
9-1	Parallel Port Electrical Specifications	9-4
10-1	RS-232C Signal Definitions	10-2
15-1	Available Device Types	15-7
15-2	GETIO Address Pointers	15-9
22-1	Memory Type	22-6
23-1	Xycom Terminal Status Codes	23-5
23-2	Omron Status Register Response Codes	23-6
23-3	Error Response Codes	23-9

WARNING

Dangerous voltages are present within the terminal. These voltages will linger after all electrical power is turned off. Use caution whenever the front panel 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.

Chapter 1

4800-Ex EXPANSION MODULES

1.1 INTRODUCTION

The 4800-Ex Series Expansion Modules are add-on modules that are installed into XYCOM expandable industrial terminals. The purpose of these modules is to give the terminal user a Screen Memory save option, battery-backed Time-Of-Day Clock, access to Xycom's own Operator Interface Language (OIL), and an additional communications port.

Three of the Expansion Modules give the user a generic additional serial RS-232, multidrop RS-485, or parallel RS-422 port.

The remaining Expansion Modules give access to the data table area on a specific of Programmable Logic Controllers (PLCs) via XYCOM's Operator Interface Language (OIL). Specifically, these modules will interface with the individual PLC or PLC network by way of a "Direct Connect". The protocol to be used by the module to communicate will depend on the particular PLC. The specific communications protocol and commands (transparent to the user) are discussed in the appropriate chapter to follow.

The 4800-Ex Modules completely alter the personality of the terminal. The command sequences and set-up menus that the terminal had used change with the installation of the module. Upon completion of the installation, the terminal will understand the English-like Operator Interface Language (OIL) and the few special command sequences it needs to exchange information with your system's processor. In addition, the terminal gains battery-backed memory, allowing permanent program storage.

Because the terminal gains a new "personality" with the expansion module, use this manual for operation details rather than the original terminal manual. **DO NOT** discard the terminal original manual -- it still contains important information.

The 4800-Ex series Expansion Modules currently available are:

- 4800-E1: Additional Serial Port
- 4800-E2: Additional Multidrop (RS-485) Port
- 4800-E3: Additional Parallel Interface (RS-422)
- 4800-E4: Additional Serial Interface (RS-232)
- 4800-E5: Additional Multidrop (RS-485)
- 4800-E6: A-B Data Highway
- 4800-E7: Modicon MODBUS

- 4800-E8: TI Series 500
- 4800-E9: Eagle Signal Controls
- 4800-E10: Square D SY/MAX
- 4800-E11: Westinghouse
- 4800-E12: GE Series 1/Series 6 and TI-305
- 4800-E13: Siemens
- 4800-E14: A-B T3
- 4800-E15: Cincinnati Millicron
- 4800-E16: TI-435

All Expansion Modules feature Screen Memory, OIL, and TOD Clock (except 48---E1 and 4800-E2).

1.2 EXPANSION MODULE OVERVIEW

The XYCOM 4800-Ex Expansion Module provides a XYCOM expandable industrial terminal with additional memory, communications and programming capabilities. The expansion board is designed for easy installation. Each expansion board is shipped with a 55K battery-backed CMOS screen program memory, which is sufficient for nearly all applications.

Screen programs (also referred to as "stored screens" or simply "screens") are a flexible means to format the screen displays (what the picture on the terminal actually looks like) for a specific application.

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

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

OIL is an easy-to-use programming language, designed to simplify the creation and enhancement of screen displays for industrial applications. OIL provides the capability 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.

The terminal 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 terminal.

A host computer can have its data displayed simply by sending new data to the terminal, which can read the new data, send it, format it, and flash it on the screen. Since the terminal 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 keypad or keyboard, 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. 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, and serial port data buffers.

With the expansion card installed, the PLC interface occurs through the option card port. This leaves the terminal serial port available for printer, tape drive, or computer link.

In some instances, the PLC requires an unsupported communication standard, or the expansion card is capable of communicating in more than one standard. In these situations a Xycom Communication Module is provided. When installed, the serial port on the option card is available for printer, tape drive, or computer link. The terminal serial port on the controller card is "dead." PLC interface is handled through the Communication Module.

1.3 SPECIFICATIONS

The following specifications are for the terminal and generic option card. Communication specifications for specific cards are given in the respective chapters.

Serial Port

- Standard RS-232C interface
- Baud rates 300, 600, 1200, 2400, 4800, 9600, 19.2K
- Full or half duplex
- Signals supported:
TD, RD, RTS, CTS
- RTS/CTS or XON/XOFF handshaking

Screen Memory

- 55K bytes standard
- Battery backed
- 255 program blocks

Time-of-Day Clock/Calendar

- Separate hour, minute, second, year, month, day, and day of week registers
- Accuracy: $\pm 0.003\%$ @ 25 °C
- Drift: -0.003% 0 °C- 50 °C
 $+0.007\%$ 25 °C- 50 °C
- Automatic leap year adjustment
- Battery-backed

Back-Up Battery

- Lithium battery
- Lithium content below FAA and ICC transport limits (.49g)
- 10 year typical life, even with full 55K memory complement

Operator Interface Language

- Built-in full-screen editor
- Immediate program execution -- no compiling
- Data transfer among:
 - host computer
 - keypad/keyboard
 - data registers
 - serial port
- Storage of all message text and attributes in terminal battery-backed memory
- Arithmetic operations (add, subtract, multiply, divide) and BCD/binary conversions on data

Operator Interface Language cont.

- 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 terminal or host computer
- Built-in diagnostic self-tests
- Program back-up and loading with tape drives or microcomputers
- Automatic start-up of user-specified program upon power-up or reset
- Help Screens

Data Registers

- 500 registers, including 10 dedicated
- Selectable placement in battery-backed memory
- 16-bit registers
- Internal and external access
- Up to 9999 extra data registers available through the configuration menu

Security

- Selectable lockout of configuration
- Optional password protection

Diagnostics

- Power-up confidence test
- Individually executable tests

Status Lines

- Menu selectable from 0 to 5 status lines

Chapter 2

4800-Ex EXPANSION MODULE INSTALLATION

2.1 INTRODUCTION

The standard (unexpanded) version of a Xycom 4800-series Industrial Terminal consists of a CRT assembly, a power supply board, and a single CRT controller board. Installation of a 4800-Ex Expansion Module is simply a matter of "piggy-backing" the Expansion Module to the existing controller board via a header plug connection and seven hex stand-off posts. Each expansion module has its own specific personality plate, which is shaped to accommodate the various port connector types found on the different expansion modules. The personality plates replace the standard (factory-installed) port cover plate on the rear of the terminal.

Beacuse of the variety of PLCs and communication standards, some of the Expansion Card kits contain a Xycom RS-232, RS-422, or 20mA Communication Adapter Module. This Adapter Module is installed to the rear wall of the terminal and connected to the controller card. This "shifts" the port functions, i.e., the port on the Adapter assumes the normal function of the port on the Expansion Card, the port on the Expansion Card assumes the function of the port on the Controller Card, and the port on the Controller Card is no longer used.

CAUTION

The XYCOM 4870 Terminal does not have space to internally mount a Communication Adapter Module.

In the case of the Xycom 4870 Flat Panel Terminal, there is no room to mount one of Xycom's Communication Adapters. The connectors are provided internally, however. The user must design a mounting for the Communication Adapter outside of the terminal, and cable it appropriately.

2.2 INSTALLATION

To install the expansion module, do the following:

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

1. Turn off power to the terminal and remove the power cord.
2. Remove the top of the terminal by loosening the six screws as shown in Figure 2-1. Save all screws and washers.

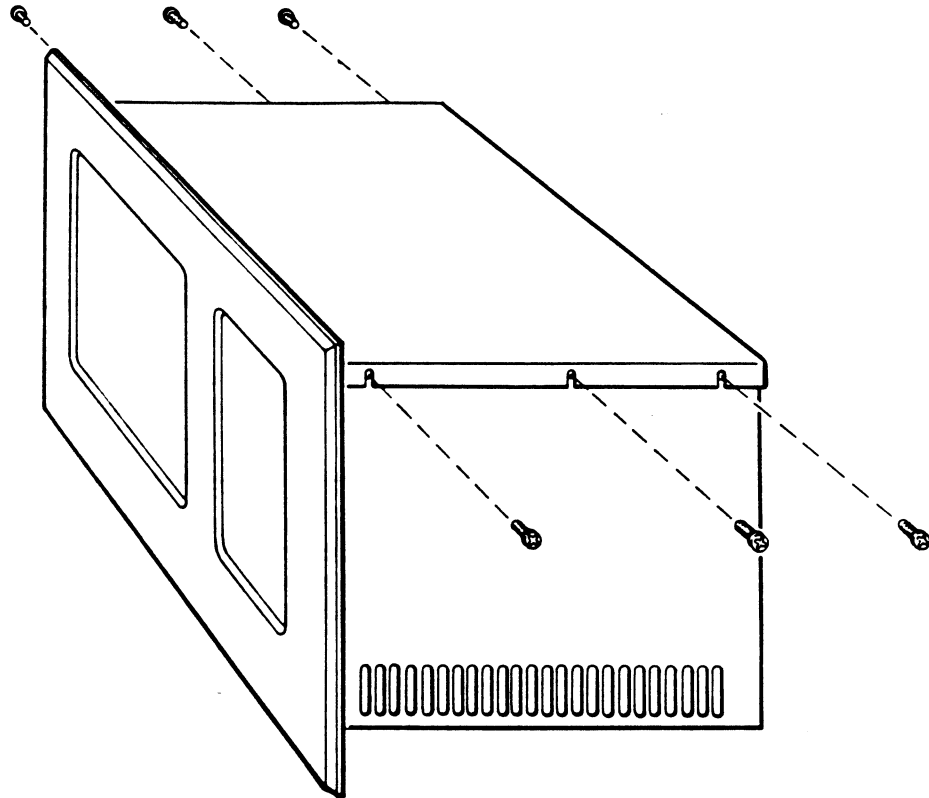


Figure 2-1 Removing the Terminal Cover

3. Locate the CRT controller board (see Figure 2-2).

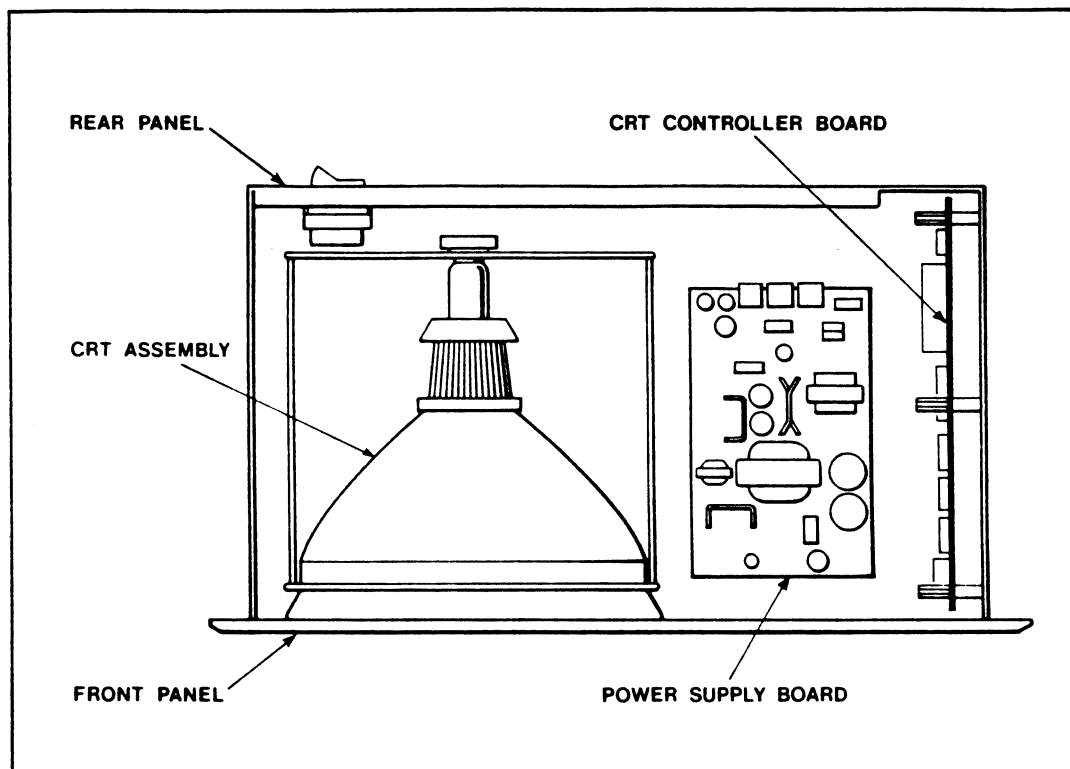


Figure 2-2 Overhead View of Terminal Components.

4. Remove the standard cover plate on the rear of the terminal (see Figure 2-3). Save all screws and washers.

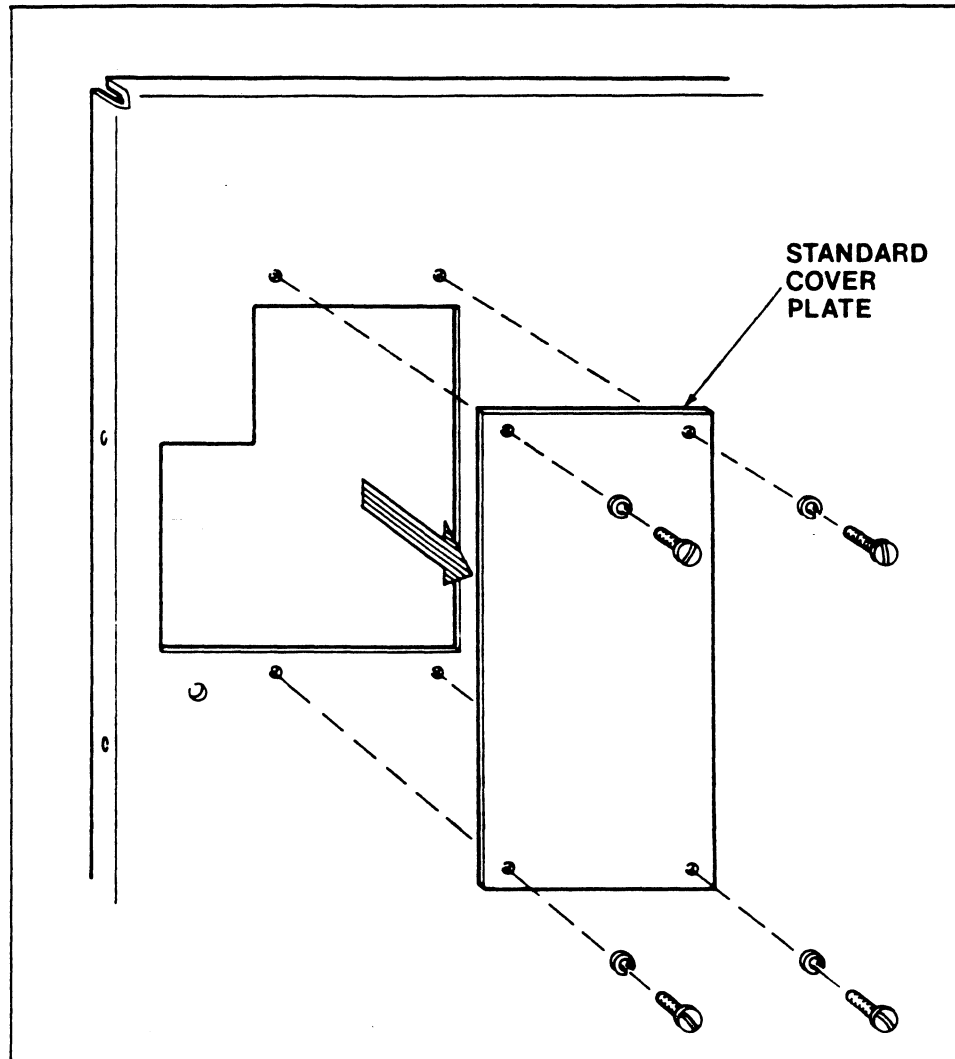


Figure 2-3 Rear Cover Plate Removal

5. Locate jumper J7 on the Expansion Module (see Figure 2-4). This jumper is used to activate the support battery for the CMOS RAM and clock/calendar. The board is shipped with the battery disconnected in order to prevent battery drain. Install the jumper at J7. The battery backup is now activated.

CAUTION
Apply even pressure as close to the header plug as possible.

6. Install the Expansion Module by lining-up the header plug on the top of the board with the corresponding header socket on the front of the CRT controller board (see Figure 2-5). Apply pressure to the Expansion Module as close to the header plug as possible. When the header pins are firmly engaged, check to see that the board is straight, and in line with the stand-off posts on the controller board.

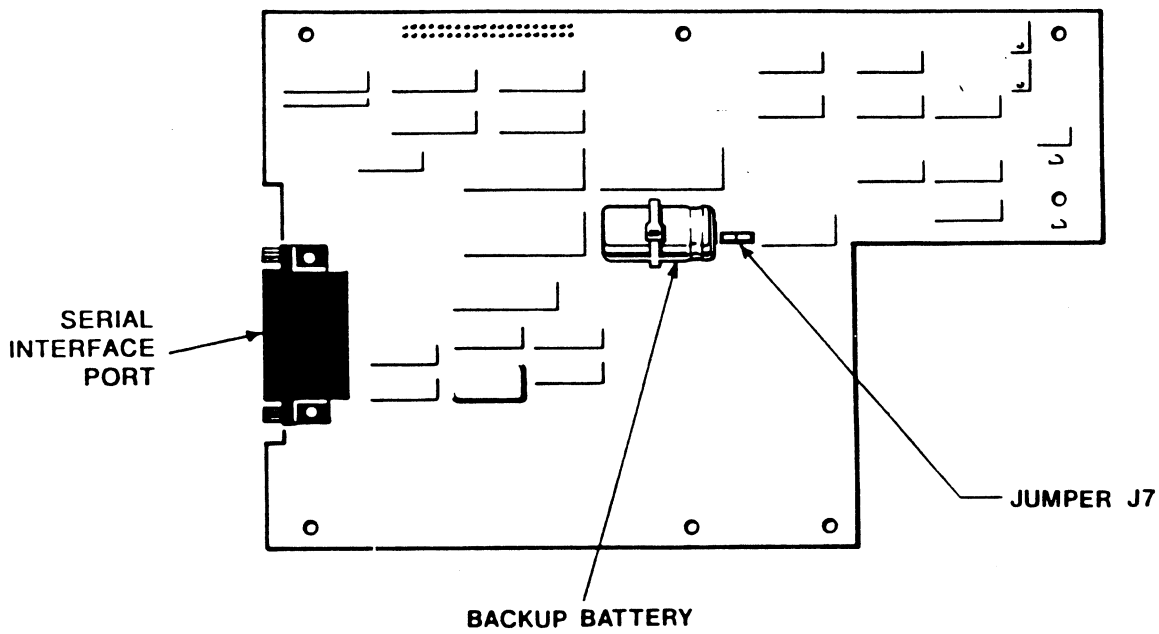


Figure 2-4 Battery Backup Activation

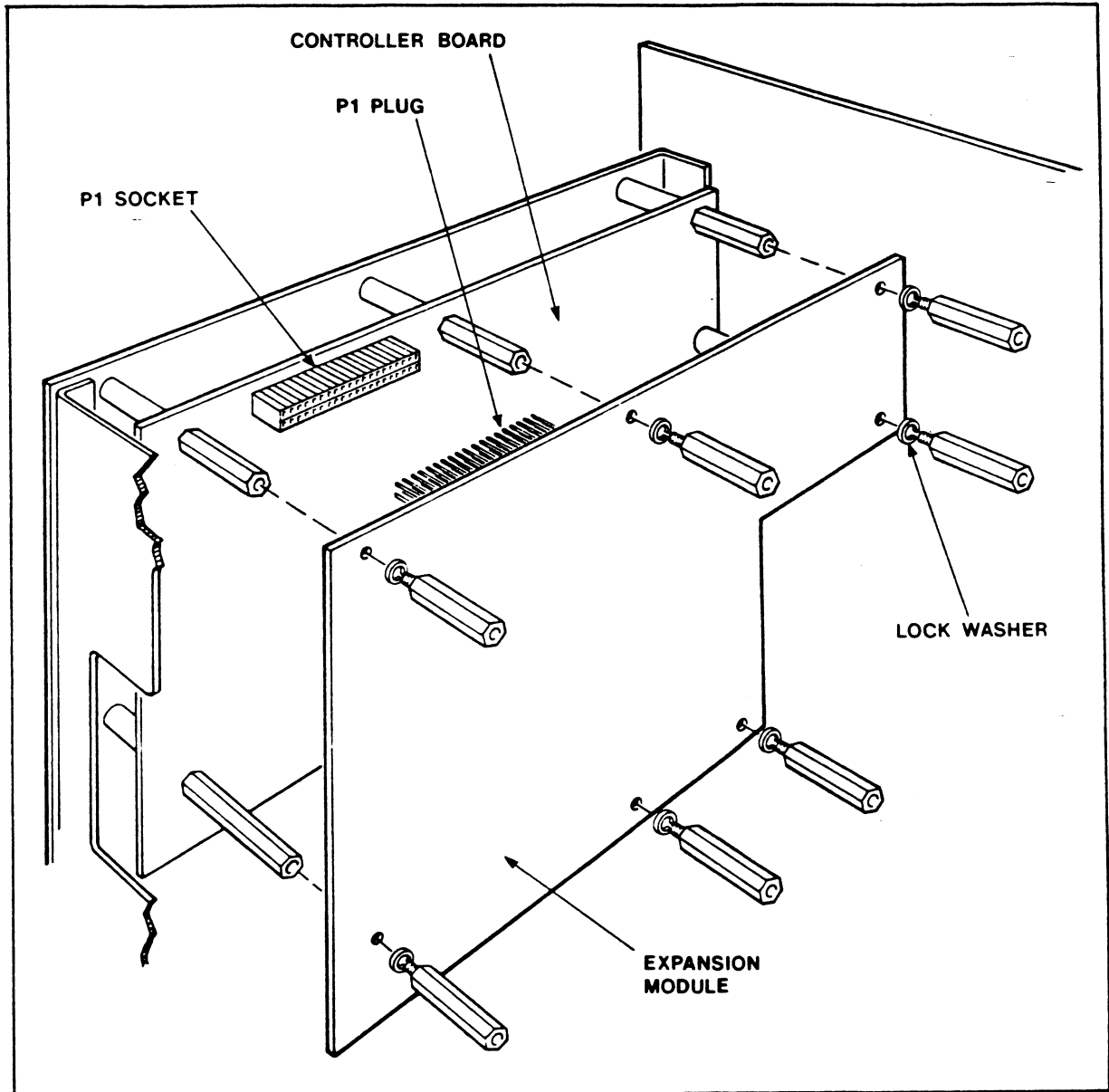


Figure 2-5 Installation of the Expansion Module

NOTE
If a Communication Adapter is to be installed, do so at this point (refer to Section 2.3).

7. Attach the Expansion Module to the CRT controller board using the additional seven stand-off posts and lock washers. The stand-off posts can be finger-tightened, and then snugged down with a hex wrench. Take care not to damage any wiring or components attached to the power supply board or the Expansion Module.
8. Install the module personality plate to the rear of the terminal (see Figure 2-6).

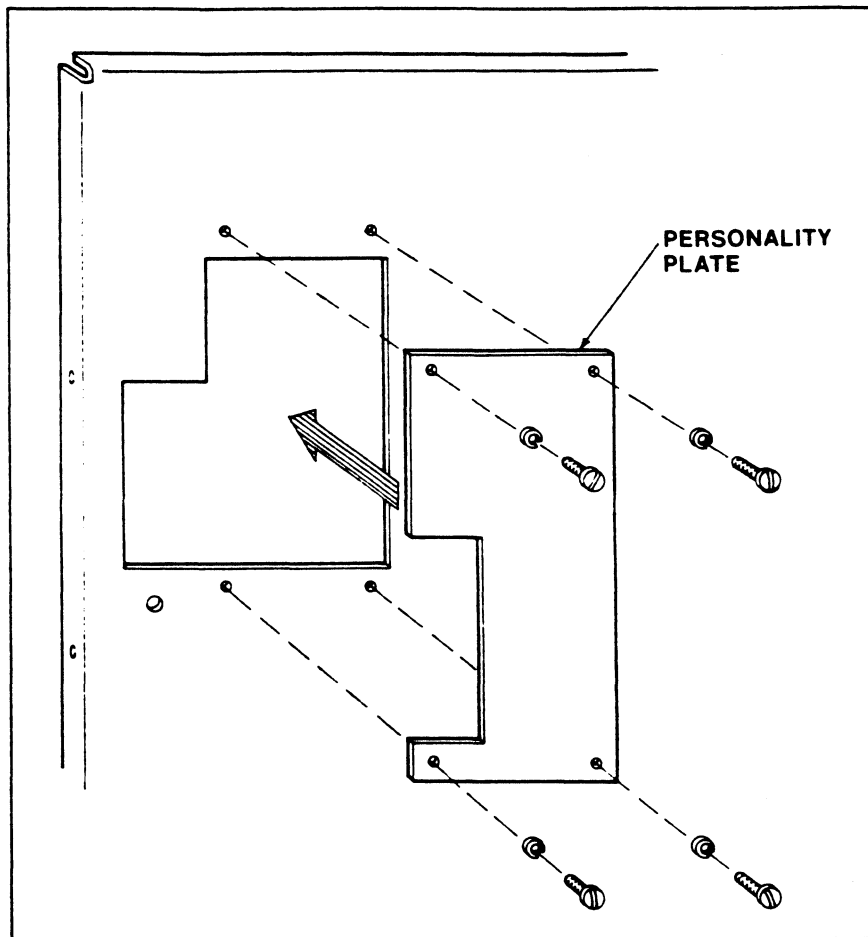


Figure 2-6 Installation of Personality Plate

9. Replace the top of the terminal.
10. Refer to the appropriate Chapter for wiring the port on the Expansion Card/Communication Adapter to the PLC.
11. Install the power cord and apply power.

2.3 COMMUNICATION MODULE INSTALLATION

The following procedure is used only if one of Xycom's Communication Modules is installed in the terminal.

CAUTION
The Xycom 4870 does not have room to internally mount a Communication Adapter Module.

1. Remove the Comm. Mod. cover plate on the rear of the terminal (see Figure 2-7). Save all screws and washers.

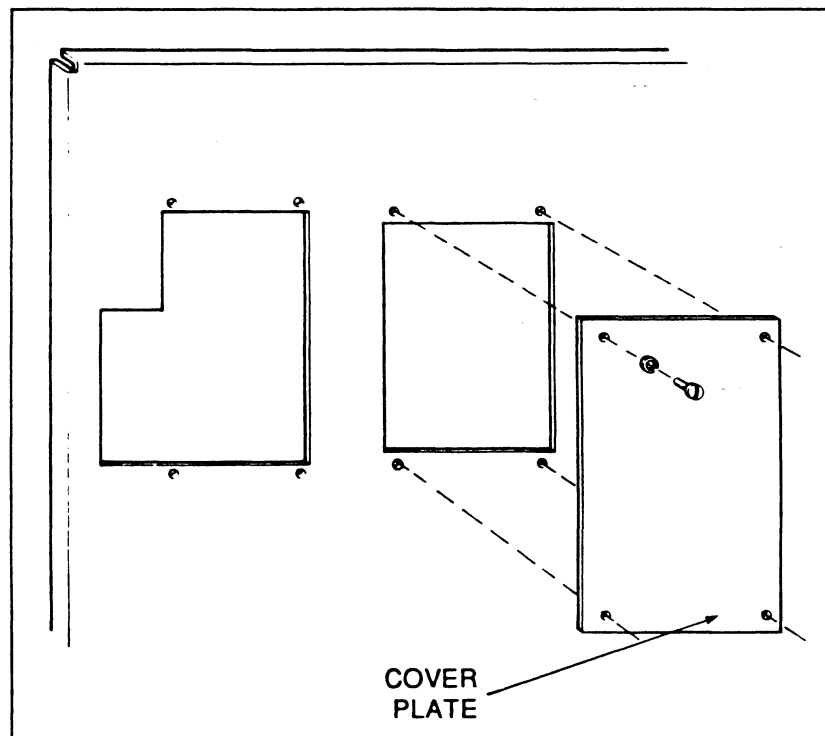


Figure 2-7 Rear Cover Plate Removal

2. Install Xycom Communication Adapter with four screws (see Figure 2-8).

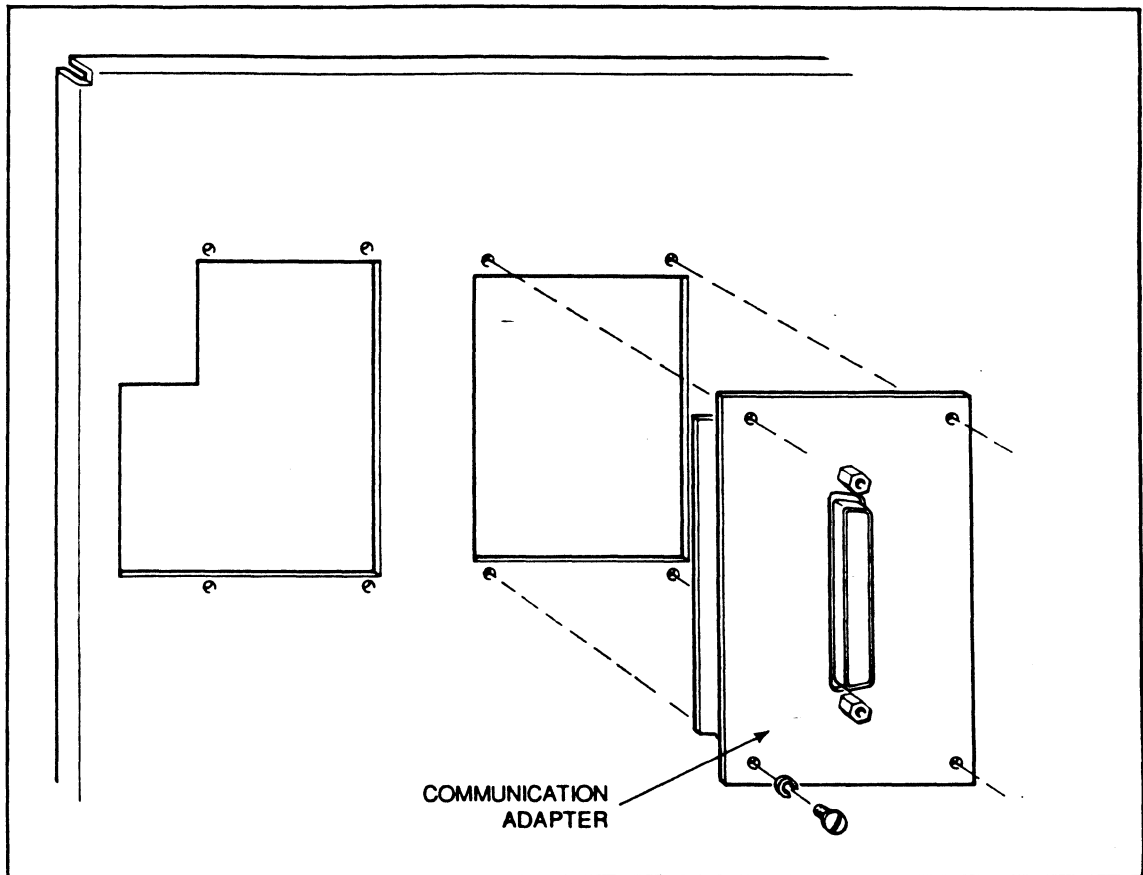


Figure 2-8 Communication Adapter Installation

3. Attach power (6 pin straight) and data (20 pin header) connectors from Communication Adapter to Controller Card (see Figure 2-9).

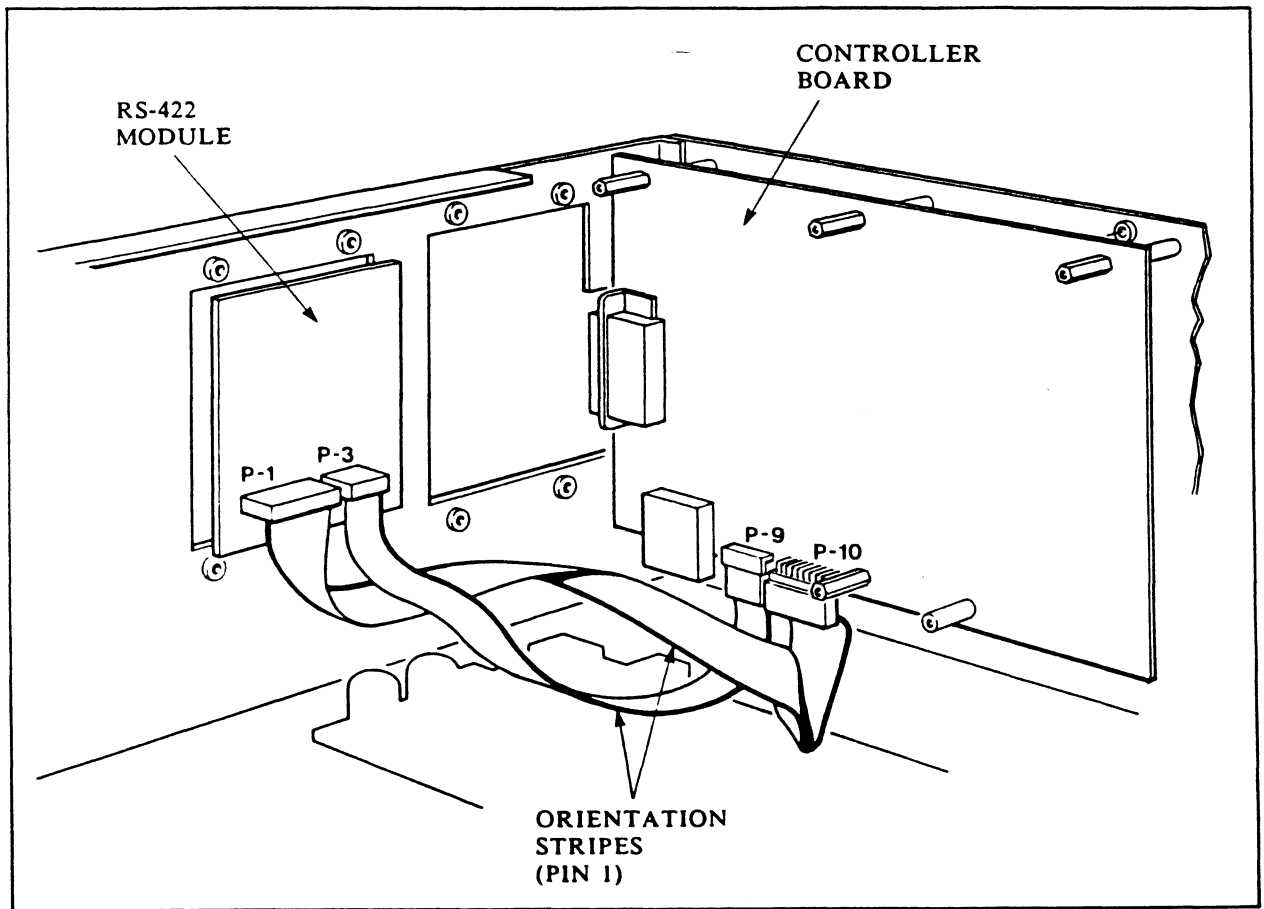


Figure 2-9 Communication Adapter Connectors

2.4 VERIFYING INSTALLATION

If the Main Menu was displayed prior to installation of the Expansion Board, it would appear like this:

```
Xycom Industrial Terminal
Release X.X

1) Configuration
2) Diagnostics
3) Set Password
4) Set Tab Stops

<RET> or <ENTER> to quit
```

Assuming that the installation of the Expansion Board was properly executed, and the board is functioning correctly, the Main Menu will now appear as follows (with the exception of the 4800-E1 and 4800-E2):

```
Xycom Intelligent Industrial Terminal
Release X.X

1) Configuration
2) Program Utilities
3) Backup/Restore
4) Diagnostics
5) Terminal Mode
6) Set Password

<RET> or <ENTER> to quit
```

If the terminal does not reflect the difference in Main Menus, go back and check the installation procedure steps in Sections 2.2 and 2.3. Please refer to Chapters 7 and 8 for information on the new menu configurations for the 4800-E1 and 4800-E2.

2.5 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, all command and data queues are cleared. However, it does not perform any diagnostic tests.

2.6 PORTS

After installation of the expansion board, the port on the Controller Card is treated as an additional serial port by the terminal. This port is available to communicate with any peripheral devices. It communicates in the RS-232C standard.

In some cases a Communication Adapter is installed to gain the correct communication standard to the PLC device and/or network. In this case, the port on the Communication Adapter is used to interface with the PLC. The port located on the Expansion Card is used for communication with any peripheral devices. The port on the terminal controller board is no longer used.

The specific configuration for the PLC "Direct Connect" is given for each in the later Chapter. Refer to the Table of Contents for a complete listing.

2.6 CONNECTING THE OPTIONAL KEYBOARD

All of the Xycom terminals will accept either an IBM PC/XT keyboard or XYCOM 4810-KYB and 4800-K1 optional keyboards. With these option keyboards a user can have the full range of ASCII & Alphanumeric characters.

Before installing the keyboard, the terminal must be configured for the type of keyboard being used. There is a switch (S1) mounted on the terminal controller board for this purpose (see Figure 2-10).

If the 4810-KYB optional keyboard is being used, then switch S1 must be set to the "Key" position. If the PC/XT keyboard is being used, then the switch must be set to the "PC" position. The 4800-K1 optional keyboard is not affected by the S1 switch.

NOTE
Both the keyboard plug and the keyboard socket are keyed to prevent incorrect installation.

To install the optional full-size keyboard, unscrew the protective cap and connect the cable from the keyboard to the port on the lower right-hand corner of the terminal front panel. When the full-size keyboard is not being used, the protective cap should be screwed on again to maintain the NEMA 4 Seal of the front panel.

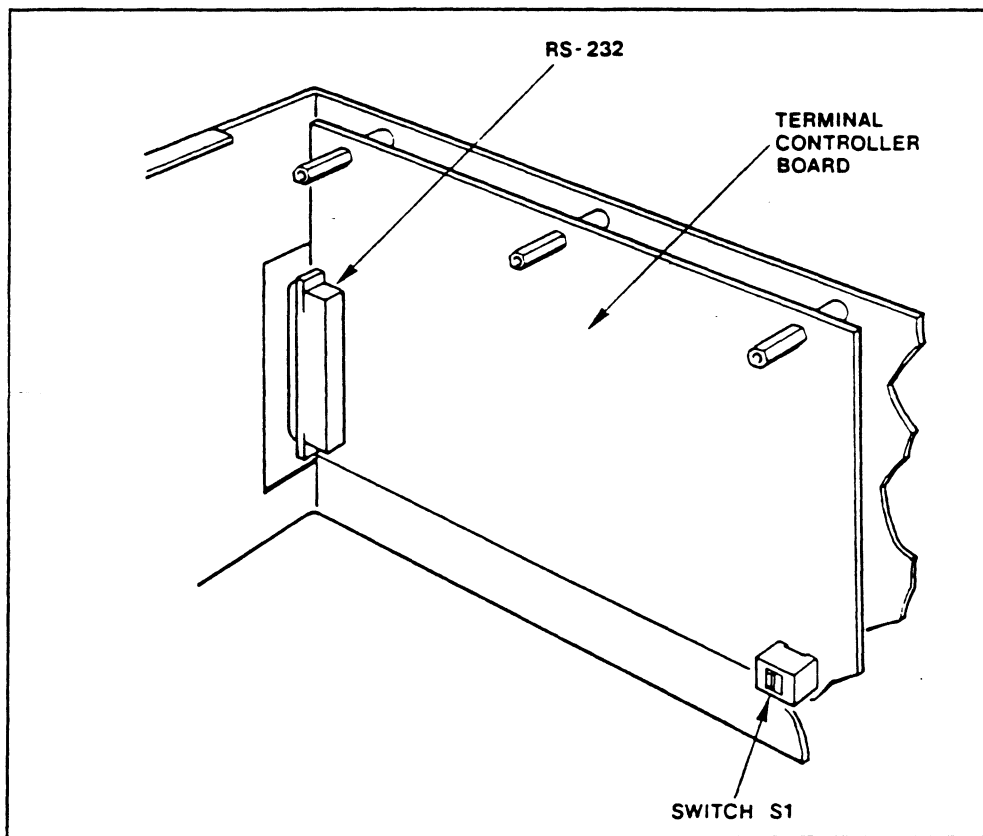


Figure 2-10 Location of Switch S1

Chapter 3
BASIC CONCEPTS

NOTE

This Chapter applies to only the 4800-Series Expansion Modules that contain OIL as an option.

3.1 INTRODUCTION

This chapter is designed to introduce the basic operational concepts which facilitate programming and application once the Expansion Module is installed in an expandable terminal.

3.2 MENU HIERARCHY

The Expansion Module allows a terminal to operate in any of three modes: "Operating Mode", "Set-Up Mode", and "Terminal Mode". It is in Operating Mode when a screen program is being executed. When in Operating Mode the terminal can execute program blocks and process remote commands. In Terminal Mode the terminal emulates a teletype or dumb terminal. When it is in Set-Up Mode, the terminal displays menus on the screen which allow the user to change its configuration, type programs into a program block, or edit existing programs.

Whenever the terminal (with the expansion board installed) is turned on, it is automatically put in the Operating Mode. To put the terminal in the Set-Up Mode do either of the following:

Press the <F10> key twice on the keyboard

or

Press the top left and lower right keys simultaneously on the front panel keypad.

NOTE

If menu lockout is set a keyboard must be used.

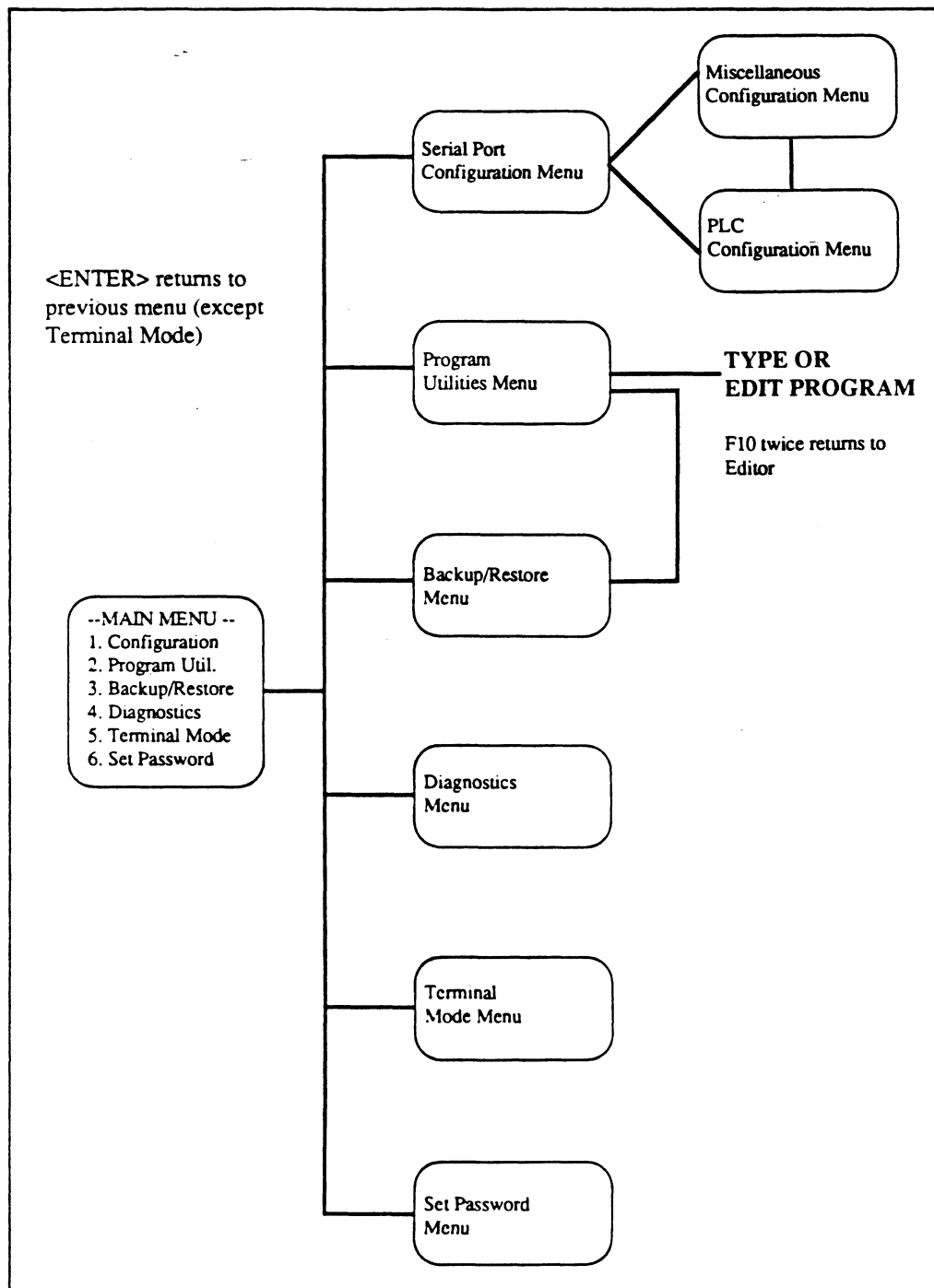


Figure 3-1 Menu Hierarchy

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

3.3 MAIN MENU

When the terminal is put into 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 |
| RET> or <ENTER> to quit |

Figure 3-2 Main Menu

Selecting any of the options (option number) will bring up secondary menus which in turn provide additional options. Figure 3-1 shows the tree structure of the option menus displayed by the terminal 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 MENU

There are three configuration menus. When you press "1" while in the Main Menu, the first configuration menu is displayed on the screen. To get to the second configuration menu, press "C" while in the first menu. To get to the third configuration menu, press "C" while in the second menu. The second menu is always for PLC interface configuration. These are discussed in the later chapter that is devoted to each Expansion Module.

3.4.1 Serial Port Configuration Menu (1st Configuration Menu)

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

-- Serial Port Configuration Menu --	
7	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
1	1=RTS/CTS Handshaking 0=XON/XOFF
0	1=Echo Input 0=Don't Echo Input
Use <UP-ARROW>, <DOWN-ARROW>, <LEFT-ARROW>, <RIGHT-ARROW>.	
Use values 0 through 9.	
"C" for next configuration menu, <RET> 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 the "ENTER" or "RETURN" key.

Baud Rate. The baud rate of the serial channel should be set to match that of the serial device to which the terminal is connected.

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

Serial Port

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

Disable or Enable Parity. Use or omission of parity bit insertion and checking can be selected. 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 terminal 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 terminal, 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 terminal 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

You can not enable handshaking and half-duplex.

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

CTS is an input to the terminal. If this line is asserted (**High**) the terminal assumes that it is time to transmit data to an external device. When CTS is inactive (**Low**) the terminal will stop transmitting data to an external device. This keeps the terminal 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 terminal when it is time for the external device to send data.

If an XON is received by the terminal, 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 terminal when the sending device should stop sending data.

If an XOFF is received by the terminal, 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.

3.4.2 Miscellaneous Configuration Menu (3rd 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" returns you to the first configuration menu.)

```

-- Miscellaneous Configuration Menu --

0      1=Block Cursor                0=Underline Cursor
0      1=Soft Scroll                 0=Pop Scroll
1      1=60 Hz.                      0=50 Hz.
0      1=Lock Keypad Menu Entry      0=Unlock Keypad Menu Entry
0      1=Data Register in CMOS       0=Data Registers in RAM
0      0=Clock Display Disabled 1=12 Hour Clock, 2=24 Hour Clock
0      Keypad - 0=4x7(A) 1=3x10(F1) 2=4x7(F1)
00     Start-up Program Block Number
0000   Number of Extra Data Registers
0      Number of Status Lines (0 - 5)
00     Year
01     Month
01     Day
00     Hour
00     Minute
00     Second

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

Figure 3-4 Miscellaneous Configuration Menu

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

Soft/Pop Scroll. Selects whether the display data moves smoothly (soft scroll) or one-line-at-a-time (pop scroll) when data moves upward ("scrolls") on the screen. Soft scroll is not available on the color terminal.

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.

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.

Data Registers in CMOS. If this field contains a 1, then the 490 user data registers are stored in battery-backed CMOS memory and are retained when the terminal is powered down. When this field is a 0, data registers are lost when power is lost, and are all cleared to zero upon terminal power-up or reset. If the data registers are stored in CMOS, 980 fewer bytes of program space is available for OIL programs.

Clock Display. This field selection allows a clock/calendar to be maintained on the 25th line of the display. The date is displayed on the left side of the last line in the following format:

DD Mon YYYY

Where DD is the day of the month, Mon is a three letter abbreviation of the name of the month, and YYYY is the year.

The time is displayed on the right side of the last line in the following format:

HH:MM:SS

Where HH is the hour, MM is the minute, and SS are the seconds. This display can be in either 12 or 24 hour format. The date/time display is updated every 29/60 seconds.

3 x 10 Keypad / 4 x 7 Keypad. This field selects which keypad is being used. The 3 x 10 key layout is found on 12" CRT terminals. The 4 x 7 key layout is found on 9" CRT terminals. The 3 x 10 keypad has the "F1" in the upper left hand corner of the keypad, and depending on the model chosen, the 4 x 7 keypad has either an "A" in the upper left hand corner or a "F1".

Start-up program block number. If this field contains a number other than 00, that particular program block will be executed whenever the terminal 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 the user to debug the start-up program block without having to turn the terminal off and on.

In order for the terminal to receive and execute commands from the control system it must be returned to Operating Mode (refer to Section 3.2.1).

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.

As with the normal registers, if there is not enough room in CMOS for the extra registers the number of extra registers is set to 0 and the user is notified, except on a power-up.

Number of Status Lines (0-5). If this menu selection is chosen, the user will have from 0 to 5 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.

Data Registers in CMOS. If this field contains a 1, then the 490 user data registers are stored in battery-backed CMOS memory and are retained when the terminal is powered down. When this field is a 0, data registers are lost when power is lost, and are all cleared to zero upon terminal power-up or reset. If the data registers are stored in CMOS, 980 fewer bytes of program space is available for OIL programs.

Year, month, day, hour, minute, seconds. Whenever the third configuration menu is called up, the current date and time is read from the clock/calendar chip and displayed. This display is not updated as the time changes, but is provided to let the user know what the date/time setting is prior to changing it.

If the user changes the values for date and time, the date/time will not be reset until the Miscellaneous Configuration screen is exited.

Whenever the terminal 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 terminal -- they can only be read. Their values can be changed only through the "Miscellaneous Configuration Menu", or through the remote command "Set Time".

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 following secondary menu will appear:

```
-- Program Utilities --  
  
1) Edit  
2) Execute  
3) Copy  
4) Backup/Restore  
5) Directory  
6) Erase  
7) Search  
  
<RET> or <ENTER> to quit
```

Figure 3-5 Program Utilities Menu

3.5.1 Using the Editor

Selecting option #1 from the Program Utilities Menu puts the terminal 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 program. There are 255 program blocks on the Expansion Module and each of them can store a separate program. In a new module, 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 terminal 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 port.

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 terminal.

Entering and editing programs requires the use of an external full-stroke keyboard (e.g., XYCOM Model 4810-KYB).

To enter a program into program block 1, type the program using the terminal were an ordinary typewriter. The <RETURN> 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. The cursor can be moved left, right, up, or down with the keys shown in Figure 3-6.

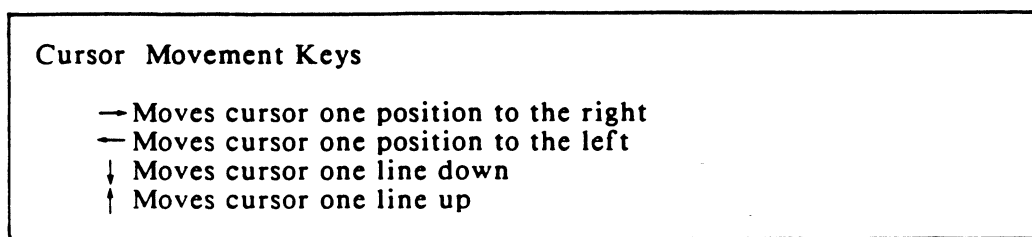


Figure 3-6 Cursor Movement Characters

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. The editor allows the deletion of the character at the cursor position or the character immediately to the left of the cursor:

Delete Characters:

 -- Deletes the character at the cursor position

<Back Space> -- Deletes the character immediately to the left of the cursor

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 <RETURN> 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 the user in editing mode. These help screens contain all the commands discussed in detail in Chapter 4 of this manual. The status line at the bottom of the edit screen displays:

Char <ALT>Line <F6>Run <F5>Quit <F1>Help

Simply press F1 for the first of a series of Help Screens. The status line now reads:

"C" for next screen, <F5> to quit

Testing a Program. After typing a program, it can be tested by immediately executing it. The program block being displayed on the screen will be executed from the beginning if the <F6> key is used. (Before executing the program, the terminal automatically clears the screen, puts the cursor in the upper left hand corner, and resets all display attributes to their default condition.)

If any typing mistakes have been made or a command incorrectly typed, the terminal will display an error message. 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 should be nearby.) To display the entire program block again, move the cursor upward to scroll the screen.

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

3.5.2 Program Execution

A program block can be executed in one of two ways:

- From within the editor, by pressing the <F6> key;
- or
- By selecting option #2 - "Execute" from the Program Utilities Menu.

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

Execute what program block number? (1-255)

The directory can be consulted first if the program block number is not known (refer to Section 3.2.3.5 on using the directory). Press <ESC> to exit the "Execute" menu without executing a program.

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

3.5.3 Copying Program Blocks

Option #3 - "Copying" from the Program Utility Menu allows the user 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 copy, 10 to 50 and 11 to 51.

NOTE

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

When copying from one program block to another, any existing program data in the destination block will be written over by the new copy. To prevent the loss of existing programs, the directory should be consulted prior to making a copy.

An easy way to delete the contents of an entire program block is to copy a blank program block into that program block. The original blank block of memory space is automatically reclaimed for further program storage.

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 magnetic tape unit or computer system. Appendix A of this manual deals with the procedure for saving and restoring programs on several different tape units. 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
- <RET> or <ENTER> to quit

The backup option saves program blocks from the screen memory to a connected back-up unit, such as a tape drive. The restore option loads previously saved programs from the tape unit into screen memory. The verify option will check data just backed-up against the original program in memory (in order to verify that the transfer was successful). Program restore/verify will indicate when the operation is complete. Refer to Appendix A for procedural information on backing-up and restoring 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 formatted for a printer. Only non-empty programs will be printed.

3.5.5 Using the Directory

Option #5 "Directory" from the Program Utilities Menu allows the user 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 semi-colon 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 F5 or CLR. Exit the directory by pressing the F5 key.

3.5.6 Erase

Option #6 - "Erase" from the Program Utility Menu allows the user 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 the user to search for a string of characters. The user 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. The user may either quit or continue searching.

3.6 BACKUP/RESTORE MENU

This option is provided as a means for saving and loading program blocks via the serial port and a magnetic tape unit. For information on using this option refer to Section 3.5.4 and Appendix A.

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.

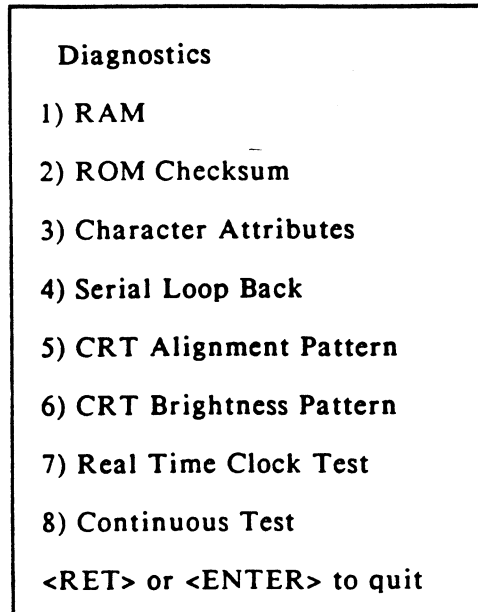


Figure 3-7 Diagnostics Menu

3.7.1 RAM Test

If the RAM test is selected, the terminal will check the CPU RAM (8031 RAM), then the external RAM, then the display RAM, and then the attribute RAM. After checking the 8031 RAM the terminal will display one of the following messages:

```

8031 RAM OK
or
8031 RAM failure

```

The next test checks the external RAM which is the serial input buffer. After testing the external RAM, the terminal displays one of these messages:

```

External RAM OK
or
External RAM failure ab/cd wxyz

```

where ab is the byte read from the failed memory address, cd is the byte that was written to the failed memory address, and wxyz is the failed memory address. All of these numbers are in hexadecimal format.

CAUTION
Turning off power while the CMOS RAM test is in progress will destroy all data in CMOS RAM.

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

CMOS RAM OK
CMOS RAM failure ab/cd wxyz

where ab is the byte read from the failed memory address, cd is the byte that was written to the failed memory address, and wxyz is the failed memory address. All of these numbers are in hexadecimal format.

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

Display RAM OK
or
Display RAM failure ab/cd wxyz

where ab is the byte read from the failed memory address, cd is the byte that was written to the failed memory address, and wxyz is the failed memory address. All of these numbers are in hexadecimal format.

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

Attribute RAM OK
or
Attribute RAM failure ab/cd wxyz

where ab is the byte read from the failed memory address, cd is the byte that was written to the failed memory address, and wxyz is the failed memory address. All of these numbers are in hexadecimal format.

3.7.2 ROM Checksum

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

3.7.3 Character Attributes Test

Characters with reverse video, high-intensity, underlining, blinking, and double-wide attributes are displayed on the status line. Color terminals display colored strips; black, blue, green, cyan, red, magenta, yellow, and white from top to bottom. Red, green, blue (background), underline, blinking, double-wide, and red, green, and blue (foreground) attributes are displayed on the status line. On a color terminal, reverse video does not apply.

3.7.4 Serial Loop Back Test

The serial ports on the terminal can be tested by selecting #4 from the diagnostic menu "Serial loop back test". This test checks first the serial port, then the PLC interface port. In order for the "loop back" test to function properly, serial ports must have certain signals looped-back for signal verification. The recommended means for looping signals is via construction of a loop-back connector using a DB-25 connector, and several jumper wires (or solder bridges). The configuration for the construction of a loop-back plug for both the PLC interface and the serial port are found in each of the Expansion Card chapters later in this manual.

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

Ctrl port: OK. xxx port: OK.

where xxx is the specific PLC port.

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

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

3.7.5 CRT Alignment Pattern

Displays alignment grid on video display until a key is pressed.

3.7.6 CRT Brightness Pattern

On the monochrome terminal, displays reverse video foreground spaces on the entire screen.

On the color terminal, displays a white screen. Displays a red screen, green screen, blue screen, black screen, and white box on a black screen, if "C" is consecutively pressed. Pressing any other key will return to the Diagnostic Menu.

3.7.7 Clock/Calendar Test

CAUTION
Turning off power while the 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 terminal will display "Clock/calendar OK" on the status line. If an error is found, the subtest which failed and error information will be displayed instead.

3.7.8 Continuous Test

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

In this mode, the terminal continuously cycles through the RAM, serial port, clock/calendar and ROM tests. If an error is found, the terminal 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.

3.8 TERMINAL MODE MENU

When this option is selected the terminal will act like a "dumb terminal" or a "teletype". Characters typed on the keyboard or keypad are sent out the serial port and echoed to the screen (if the terminal 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, or by simultaneously pressing the top left and bottom right keys on the keypad (unless keypad menu lockout is enabled).

3.9 SET PASSWORD MENU

The terminal provides two ways of "tamper-proofing" the terminal's configuration: a password and a keypad menu lockout.

Password. Whenever exiting the Operating Mode (see the previous section), if a password has been selected the following prompt will be displayed:

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

The password consists of 3 alphanumeric characters. Type the correct 3-character sequence and the Main Menu of the Set-up Mode will be displayed. Type an incorrect password and the terminal remains in the Operating Mode.

The password can be changed or disabled by selecting item 6 from the Main Menu. (To change or disable the password the Main Menu must be accessed, which requires the current password.) In response to selecting item 6, the following prompt will be displayed:

Enter new password

The user then has the following options:

- To change the password, type any three alphanumeric characters on the keyboard, then press the <Return> key. This will be the new password.
- To disable the password, just press the <ESC> key on the keyboard, without typing any characters. (The password can be subsequently reenabled by reselecting item 6 from the Main Menu.)
- To not change the password, just press the <RET> or <ENTER> key without typing any characters.

In case the user forgets the password, the remote command Return Password will return the password to the host computer (see Chapter 6).

Keypad Menu Lockout. There is an option in the Configuration Menu to lock out menu entry from the keypad. If the keypad is locked out, the password prompt can be invoked only by pressing <F10> twice in succession on the full-size keyboard. This has the effect of preventing the user from entering Set-up Mode from the keypad.

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 subsequent chapters on each module. This chapter has been designed to be both a tutorial and an in-depth reference guide. 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	<data register>	4.6.3
Position	@[<xcoord>],[<ycoord>]	4.6.4
Label	%<label>	4.6.5
Add to Register	ADD,<numerical argument>,<data register>	4.6.6
AND	AND,<data register>,<numerical argument>	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,<data register>,<bit>	4.6.12
Clear Screen	CS	4.6.13
Clear Window	CW	4.6.14
Down	D[,<numerical argument>]	4.6.15
Decrement Register	DEC,<data register>	4.6.16
Divide Register	DIV,<data register>,<numerical argument>	4.6.17
Execute Sub-Program Block	ES,<numerical argument>	4.6.18
Execute Program Block	EX,<numerical argument>	4.6.19
Exit	EXIT	4.6.20
Draw Filled Box	FBX,<bval>,[<xstart>],[<ystart>],[<xend>],[<yend>]	4.6.21
Go	GO,<label>	4.6.25
Horizontal Bar Left	HBL,[<xstart>],[<ystart>],<length>,<max length>	4.6.26
Horizontal Bar Right	HBR,[<xstart>],[<ystart>],<length>,<max length>	4.6.27
Draw Horizontal Line	HL,<lval>,[<xstart>],[<ystart>],<hlength>	4.6.28

Table 4-1 OIL Commands by Section cont.

COMMAND	SYNTAX*	SECTION
If	IF,<condition>:<command>	4.6.29
If Bit	IFB,<data register>,<bit>:<command>	4.6.30
Increment Register	INC,<data register>	4.6.31
KEXIT	KEXIT	4.6.32
Keypad Status	KS,<data register>	4.6.33
Left	L[,<numerical argument>]	4.6.34
Remainder (Modulus)	MOD,<data register>,<numerical argument>	4.6.35
Multiply Register	MUL,<data register>,<numerical argument>	4.6.36
New Line	NL[,<numerical argument>]	4.6.37
NOT	NOT,<data register>	4.6.38
ONKEY	ONKEY,<numerical argument>	4.6.39
OR	OR,<data register>,<numerical argument>	4.6.40
Plot	PLOT,<xpoint>,<ypoint>	4.6.41
Pause	PS,<numerical argument>	4.6.42
Right	R[,<numerical argument>]	4.6.46
Reset Attributes	RE,{<attribute>}	4.6.47
Restore Attributes	RSTA,<data register>	4.6.48
Save Attributes	SAVA,<data register>	4.6.49
Set Background Color**	SBC,<color>	4.6.50
Set Attributes	SE,{<attribute>}	4.6.51
Set Bit	SETB,<data register>,<bit>	4.6.52
Set Foreground Color**	SFC,<color>	4.6.53
Stop	STOP	4.6.54
Subtract from Register	SUB,<numerical argument>,<data register>	4.6.55
Transfer	TR,<source>,<destination>	4.6.56
Up	U[,<numerical argument>]	4.6.57
Unplot	UNPLOT,<xpoint>,<ypoint>	4.6.58
Vertical Bar Up	VB VBU,[<xstart>],[<ystart>],<new height>,<max height>	4.6.59
Vertical Bar Down	VBD,[<xstart>],[<ystart>],<new height>,<max height>	4.6.60
Draw Vertical Line	VL,<lval>,[<xstart>],[<ystart>],<vlength>	4.6.61
XOR	XOR,<data register>,<numerical argument>	4.6.62

*See the next page for the meaning of the symbols used here.

**Color terminals only.

4.2 COMMAND LIST

Table 4-2 lists all the available OIL commands and their formats. The OIL commands and Section numbers are listed in table 4-2.

Table 4-2 OIL Commands by Function

COMMAND	SYNTAX*	SECTION
Screen Text	"{character}"	4.6.2
Register Value Display	<data register>	4.6.3
select character size and attributes		
attributes (i.e., blinking or reverse video)		
Set Attributes	SE,{<attribute>}	4.6.49
Reset Attributes	RE,{<attribute>}	4.6.45
Save Attributes	SAVA,<data register>	4.6.47
Restore Attributes	RSTA,<data register>	4.6.46
color commands (Color Terminal only)		
Set Foreground Color**	SFC,<color>	4.6.51
Set Background Color**	SBC,<color>	4.6.48
move the cursor		
Up	U[,<numerical argument>]	4.6.55
Down	D[,<numerical argument>]	4.6.15
Left	L[,<numerical argument>]	4.6.33
Right	R[,<numerical argument>]	4.6.44
Position	@[<xcoord>],[<ycoord>]	4.6.4
New Line	NL[,<numerical argument>]	4.6.36
draw figures and lines		
Draw Box	BX,<bval>,[<xstart>],[<ystart>],[<xend>],[<yend>]	4.6.9
Draw Filled Box	FBX,<bval>,[<xstart>],[<ystart>],[<xend>],[<yend>]	4.6.21
Draw Vertical Line	VL,<lval>,[<xstart>],[<ystart>],[<vlength>]	4.6.59
Draw Horizontal Line	HL,<lval>,[<xstart>],[<ystart>],[<hlength>]	4.6.28
Vertical Bar Up	VB VBU,[<xstart>],[<ystart>],[<new height>],[<max height>]	4.6.57
Vertical Bar Down	VBD,[<xstart>],[<ystart>],[<new height>],[<max height>]	4.6.58
Horizontal Bar Right	HBR,[<xstart>],[<ystart>],[<new length>],[<max length>]	4.6.27
Horizontal Bar Left	HBL,[<xstart>],[<ystart>],[<new length>],[<max length>]	4.6.26
Plot	PLOT,<xpoint>,<ypoint>	4.6.39
Unplot	UNPLOT,<xpoint>,<ypoint>	4.6.56

Table 4-2 OIL Commands by Function cont.

COMMAND	SYNTAX*	SECTION
move data between the keyboard, data registers, and host computer		
Transfer	TR,<source>,<destination>	4.6.54
conditional and looping commands		
If	IF,<condition>:<command>	4.6.29
If Bit	IFB,<data register>,<bit>:<command>	4.6.30
Go	GO,<label>	4.6.25
Label	%<label>	4.6.5
set and clear bits of a data register		
Set Bit	SETB,<data register>,<bit>	4.6.50
Clear Bit	CLRB,<data register>,<bit>	4.6.12
arithmetic operations		
Increment Register	INC,<data register>	4.6.31
Decrement Register	DEC,<data register>	4.6.16
Add to Register	ADD,<numerical argument>,<data register>	4.6.6
Subtract from Register	SUB,<numerical argument>,<data register>	4.6.53
Multiply Register	MUL,<data register>,<numerical argument>	4.6.35
Divide Register	DIV,<data register>,<numerical argument>	4.6.17
Remainder (Modulus)	MOD,<data register>,<numerical argument>	4.6.34
logical operations		
AND	AND,<data register>,<numerical argument>	4.6.7
XOR	XOR,<data register>,<numerical argument>	4.6.60
OR	OR,<data register>,<numerical argument>	4.6.38
NOT	NOT,<data register>	4.6.37
program nesting		
Execute Program Block	EX,<numerical argument>	4.6.19
Execute Sub-Program Block	ES,<numerical argument>	4.6.18
Exit	EXIT	4.6.20
Stop	STOP	4.6.52

Table 4-2 OIL Commands by Function cont.

COMMAND	SYNTAX*	SECTION
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
Keypad Status	KS,<data register>	4.6.32
Pause	PS,<numerical argument>	4.6.40
ONKEY	ONKEY,<numerical argument>	4.6.39
KEXIT	KEXIT	4.6.32

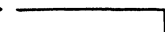


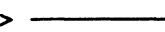
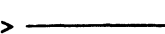
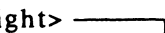

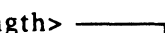
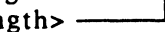
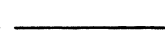





*See below for the meaning of the symbols used here.

**Color terminals only.

KEY

<u>SYMBOL</u>	<u>MEANING</u>
[] _____	Optional argument
() _____	One or more occurrences of this argument can be included
_____	Choice of either argument
<character> _____	Any single character (e.g., N, m, \$)
<data register> _____	#n or \$n, where n=1 through 500
<bit> _____	A specific bit within a register (0-15)
<numerical argument> _____	A number (0-65535), data register, or text argument
<text argument> _____	One or two characters enclosed in double quotes (c.g., "A" or "ra")
<xcoord> _____ <xstart> _____ <xend> _____	Column coordinate (0 - 79)

KEY cont.

<u>SYMBOL</u>	<u>MEANING</u>
<ycoord> 	Row coordinate (0 - 23)
<ystart> 	
<yend> 	
<vlength> 	Vertical length of graphic (0 - 24)
<hlength> 	Horizontal length of graphic (0 - 80)
<new height> 	Height of graphic (0 - 255, 0 - 240 for color)
<max height> 	
<new length> 	Length of graphic (0 - 255)
<max length> 	
<xpoint> 	Column coordinate (0 - 159)
<ypoint> 	Row coordinate (0 - 71, 0 - 47 for color)
<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>
<destination> 	Destination for data in a transfer. Any of the following: SC (screen output) SO (serial output) NO (dummy output) \$n <data register>

KEY cont.

SYMBOL	MEANING
<attribute>	Character attribute. Any of the following: QS (quad-size) DS (double size) RV (reverse video) monochrome only HI (highlight) monochrome only 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)
<condition>	A logical function inserted between two numerical arguments It can be: <, >, <>, =, =>, >=, <=, or =<
<label>	A sequence of alphanumeric characters
<color>	A field color. It can be: BLK, BLU, GRN, CYN, RED, MAG, YEL, or WHT or 0, 1, 2, 3, 4, 5, 6, or 7
<device>	A device used to handle data. It can be: KB (keyboard or keypad) SI (serial input) SO (serial output)
<addr>	Address from which data is to be read
<dest>	Address of remote PLC device.
<lngth>	Number of words (16-bit values) to read, starting at addr.
<dreg>	Destination Register number to store received data into.
<creg>	Communication Status Register.
<sreg>	Source Register number to send data from.

4.3 OVERVIEW OF THE OPERATOR INTERFACE LANGUAGE (OIL)

A program consists of sequences of displayable characters combined with commands from the 4800-E10 terminal in OIL. 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 commands from the OIL. The 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 short examples have included commands which draw figures (Draw Box), move the cursor (Cursor Position), and set character attributes (Set Attributes, Reset Attributes).

The terminal can create more complicated screen displays, including:

- nested displays in which one program block executes others (ES 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 TERMINAL

An expandable terminal with the Expansion Module installed 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 port, data registers, PLC data registers, and the 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 terminal.

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 serial port by a simple command. This program block can perform functions such as transmitting data out the serial port. In this way the user can transmit data from the keyboard to another device. Or any device can use the terminal to output data serially. Since the terminal 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 terminal can transmit specific data and/or character strings over the serial port.

A main reason for the versatility of the terminal is the Transfer command (TR). This command can move data from any of the following sources:

- Serial input port
- Keyboard and Keypad
- Data Registers

to any of the following destinations:

- Terminal Screen
- Serial output port
- Data Registers

The Transfer command also has the powerful ability to 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.

Another reason for the versatility of the expanded terminal is that when it is in operating mode it constantly monitors the serial port for data, even while it is executing a program block. The terminal stores received data in large queues which are associated with the serial port. This data will be stored in the queue until a program block reads the serial input port queue.

The expanded terminal 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. This allows the host computer to concentrate on the logic of controlling the machine or process, while the terminal concentrates on the logic of interacting with the human operator of the system.

4.4.1 Data Registers

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

These registers are no different than the 500 standard data registers except that they will always reside in CMOS RAM. The terminal 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).

Data registers can be kept either in non-volatile memory (battery-backed CMOS RAM) or in volatile memory, as selected from the Setup menu. If the registers are kept in non-volatile memory, their contents will remain unchanged when the terminal is reset, goes through a power-down/power-up cycle, or a program block is executed from the editor or the Program Utilities menu.

Choosing non-volatile data registers removes about 1000 bytes of CMOS RAM storage that would otherwise be available for OIL program blocks. If the registers are kept in volatile memory, they are initialized to zero upon reset or power-up.

Registers #1 through #10 are special-purpose registers. Registers #1 through #7 contain the time and date, and are automatically updated by the terminal. Registers #8 through #10 hold the number of characters currently in the keyboard queue and serial input queues. Registers #1 through #10 cannot be altered by an OIL program running on the terminal. Transferring data into these registers will not alter their values.

4.4.2 Transferring Data to the Terminal

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

There are two queues for the serial port. The command queue and the data queue. All information coming into the serial port 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, #11 - #500. This command is described in detail in Chapter 6.

4.4.3 Requesting Data from the Terminal

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 terminal will transmit the contents of the specified data register to the port. (See Chapter 6 for details.)

4.4.4 Transmitting Data from the Terminal

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.

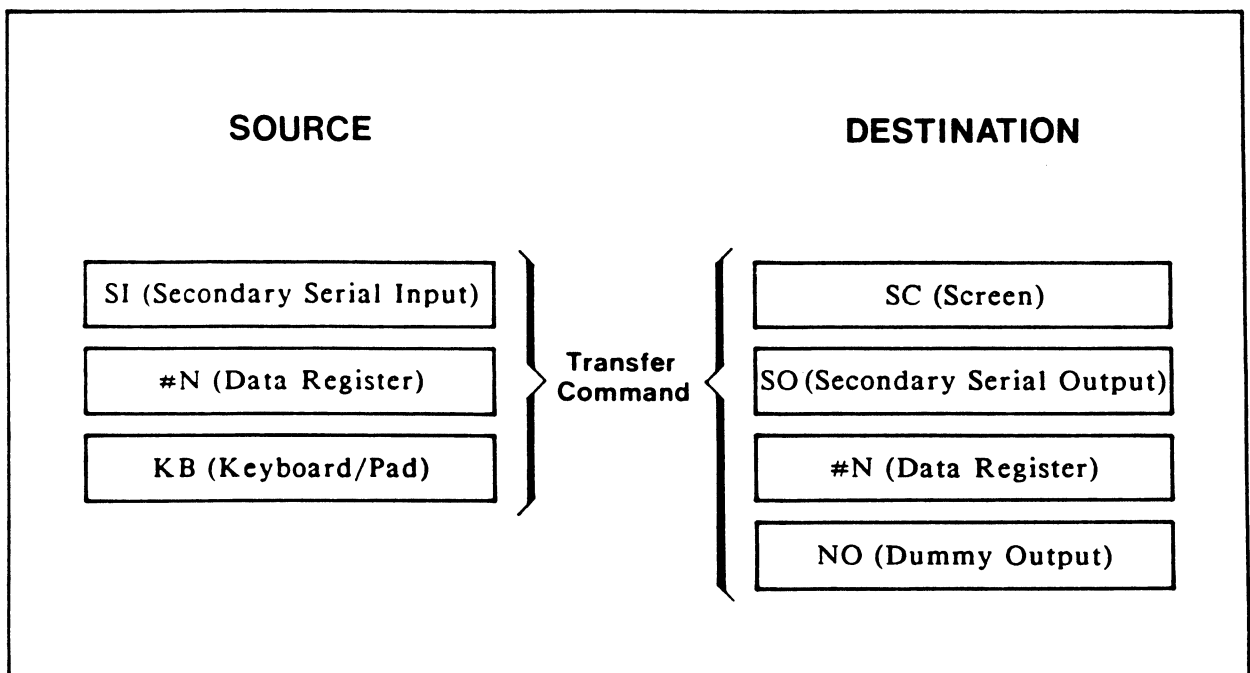


Figure 4-1 The Block Command Transfer (TR)

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 RS-422 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 4.6.53.1.

TR SI(picture) SC(picture)

Whatever is received at the serial port will be displayed on the terminal 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 4.6.53.

4.5 PROGRAMMING RULES

Commands can be typed in either upper case or lower case (or a mixture of the two).

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,bl,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. (Of course, if a semicolon is inside double quotes, it is regarded as a literal.) 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 suggested it be 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 terminal 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 inter-changeable as are the direct register (#) and indirect register (\$), for more information please refer to Section 5.2, Data Registers.

4.5.1 Communication Status Register

This is a dedicated register that monitors the data transfer operations in the terminal/PLC interface. The user can get this status for each data transfer by reading this register. Each individual expansion module from the 4800-E6 and up have their own specific Communication Status Register formats. Please see the appropriate Chapter for details.

4.6 COMMANDS

The commands are listed in the same order as found at the beginning of this chapter which may be used as an index to the commands.

4.6.1 User Interface Considerations

The user interface gives the user the ability to monitor or change areas of the network devices via the OIL command language. The way the user "gets at" the device is different for each PLC direct connect interface. The protocols are specified in the appropriate chapter later in this manual.

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 Display Register Value (# or \$)

Syntax:

#n
\$n

where:

n is a register number (1-500)

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.

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

Registers printed by this method are:

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

Example 2:

Register	Printed
05000	5000_
00050	50_
00000	0_

where: _ = space

4.6.4 Position Cursor (@)

Syntax:

@x,y

moves the cursor to a specified position on the screen. The first coordinate (X) specifies the column, the second coordinate (Y) specifies the row. The column positions range from 0 through 79, the row positions from 0 to 23.

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. 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.

A label can appear anywhere in a line. For example, "This is a test":NL:%start:"Hello":NL:go,start.

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
```

Example:

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

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

4.6.6 Add to Register (ADD)

Syntax:

ADD,num,reg

where:

num is a numeric value (0 - 65535)

reg is a register (1 - 500)

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. A register can be specified as the m value. For example, if register #100 contains the number 5 and register #11 the value 21, after this command:

AD, #100,#11

register #11 will contain the value 26.

Example:

ADD,100,#46

This example will add 100 to Register #46.

4.6.7 AND

Syntax:

AND,reg,num

where:

reg is a register (1 - 500)

num is a numeric value (0 - 65535)

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. This command can be used to force particular data bits to zero values.

Example:

AND,#20,31

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

4.6.8 Beep (BP)

Syntax:

BP

Causes the terminal 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, e.g. ">") or number corresponding to a character = 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 2:

BX,1,,,30,20

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

Example 3:

BX,2,20,10,,

This example will create a box whose bottom right 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

SI is serial input

SO is serial output

The Clear Buffer command (CB) empties one of the device buffers. The allowable parameters are KB (Keyboard), SI (serial input), and SO (serial output). For example, CB,KB clears the Keyboard queue.

Example:

CB,KB

4800-Ex Manual
January, 1990

4.6.11 Clear Line (CL)

Syntax:

CL

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 (1 - 500)

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.

4800-Ex Manual
January, 1990

4.6.13 Clear Screen (CS)

Syntax:

CS

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

Clears either the scrolled or non-scrolled portion of the screen, depending on the current position of the cursor. A CW command performed in the non-scrolled area sets the colors for the clock display.

Example:

CW

4.6.15 Down (D)

Syntax:

D,num

where:

num is a number (0 - 65535)

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 terminal manual for a description of cursor movement with large characters.

Example:

D,#11

This example moves the cursor down the number of lines specified in register ;#11

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.

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

Example:

DEC,#47

This example will decrement register 47 by one.

4.6.17 Divide Register (DIV)

Syntax:

DIV,reg,num

where:

n is a register (1 - 500)

m is a numeric value (0 - 65535)

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

In DIV the m value cannot be zero and must be less than 256.

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 following value:

```
#80 -- 3
```

4.6.18 Execute Sub-program Block (ES)

Syntax:

ES,block

where:

block is the number of the next sub-program block to be executed (1 - 255)

The ES command causes the terminal to execute another block, then return to finish execution of the current one. (Note that the EX 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

Then, after command 2 has been executed, execution of command block 34 will begin. After execution of block 34 is completed, execution of program block 20 will resume with command 3.

It is called "nesting" when program blocks contain ES commands. Sub-programs may be nested ten programs deep. The next chapter provides a more detailed description of nested programs.

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.

ER)

4.6.19 Execute Program Block (EX)

Syntax:

EX,block

where:

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

The EX command causes another program block to be executed. Unlike the ES command, control does not return to any statements following the EX command.

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, program block 121 can contain "EX 121". In that case, it would be re-executed -- ad infinitum.

Example 1:

EX,230

This example executes program block 230.

Example 1:

EX,#68

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

4.6.20 Exit from Program Block (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

4.6.21 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, the xstart = xend and ystart = yend. This results in boxes that are 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.

4.6.22 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. Example:

```
%label:"ABC"  
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 -- endlessly.

4.6.23 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 length of the bar, in pixels

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). 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

* The character cells are defined as:

Monochrome terminal: each character cell is 5 units wide (1 unit = 1/5 character cell)

Color terminal: each character cell is 8 units wide (1 unit = 1/8 character cell)

Flat Panel terminal: each character cell is 8 units wide (1 unit = 1/8 character cell)

4.6.24 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 the length of the bar, in units of a character cell

Draws a bar of length <new-length> 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 <max-length> is erased. The parameters <new-length> and <max-length> 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 terminal, <new-length> would be 33. The parameter <max-length> specifies how long a bar to erase before the bar of <new-length> is drawn. Therefore, <max-length> should be no less than <new-length>.

The maximum length for <new-length> and <max-length> is 255 for monochrome, color, and flat panel. 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.

* The character cells are defined as:

Monochrome terminal: each character cell is 5 units wide (1 unit = 1/5 character cell)

Color terminal: each character cell is 8 units wide (1 unit = 1/8 character cell)

Flat Panel terminal: each character cell is 8 units wide (1 unit = 1/8 character cell)

4.6.25 Draw Horizontal Line (HL)

Syntax:

HL,lval,xstart,ystart,hlength

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

hlength the number of characters the line extends to the right

Draws a line of length <hlength> 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 ">".

4.6.26 CONDITIONAL (IF)

Syntax:

IF,aregRbreg: command

- or -

IF,aregRx: command

- or -

IF,xRy: command

where:

areg and breg are two registers whose values are being compared

R 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

x,y are any numerical value (0 - 6335)

"command" is any of the other OIL commands

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.

4.6.27 IF Bit (IFB)

Syntax:

IFB,reg,bit

where:

reg specifies a register (1-500)

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 #0 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"
EXIT
%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 "Hello" in a thick-line box. If they are not equal, it will print the message "Goodbye" in a thin-line box.

4.6.28 Increment Data Register (INC)

Syntax:

INC,reg

where:

reg is the register to be modified (1 - 500)

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

4.6.29 Key Exit (KEXIT)

Syntax:

KEXIT

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

NOTE

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

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" is selected) or lower case (if "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 4.6.36).

4.6.30 Keypad Status (KS)

Syntax:

KS,reg

where:

reg is the register to place the keypad status into.

The Keypad Status command allows the user 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 terminal, not for the external serial or matrix keyboard.

Example:

KS,#20

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

4.6.31 Left (L)

Syntax:

L,num

where:

num is a number (0 - 65535)

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 your terminal 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.

4.6.32 Remainder (MOD)

Syntax:

MOD,reg,num

where:

reg is a register (1 - 500)

num is a numeric value (0 - 65535)

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
```

4.6.33 Multiply Register (MUL)

Syntax:

MUL,reg,num

where:

reg is a register (1 - 500)

num is a numeric value (0 - 65535)

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.

4.6.34 New Line (NL)

Syntax:

NL,num

where:

num is a number (0 - 65535)

If no number is specified, moves the cursor to the beginning of the next line. If a numerical argument n is specified, moves the cursor down n 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 terminal 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).

4.6.35 NOT

Syntax:

NOT,reg

where:

reg is a register (1 - 500)

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.

<p style="text-align: center;">NOTE NOT has only one register argument.</p>
--

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.

4.6.36 ONKEY

Syntax:

ONKEY,block

where:

block is the number of a 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 not 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:

```
;Example use of ONKEY - 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 4.6.29).

4.6.37 OR

Syntax:

OR,reg,num

where:

reg is a register (1 - 500)

num is a numeric value (0 - 65535)

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 Reg. 20 with the number 31 and the result is placed in Reg. 20.

4.6.38 Plot (Plot)

Syntax:

PLOT xpoint,ypoint

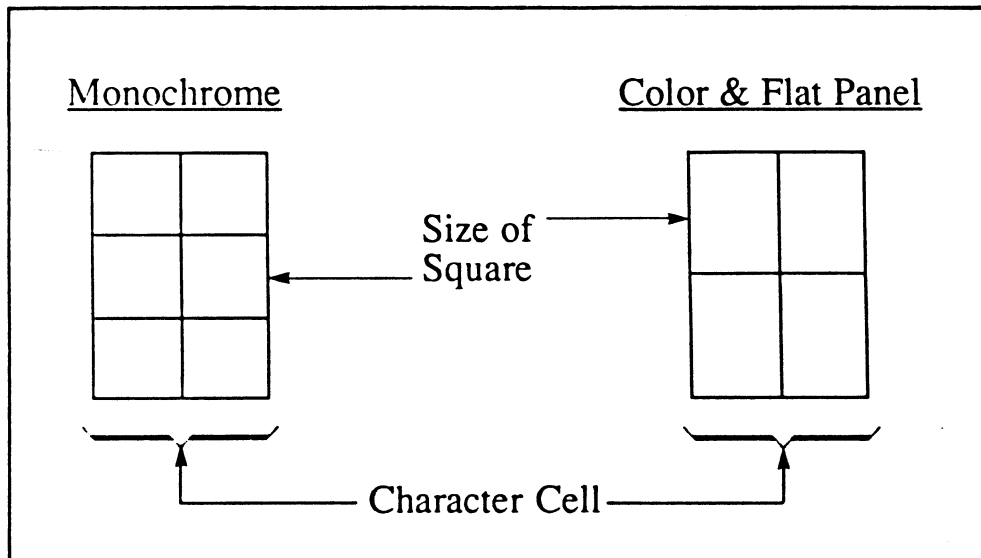
where:

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

ypoint is the row coordinate of the point (0-71 for monochrome, 0-47 for color and flat panel).

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 for monochrome terminals, (1/4 for color and flat panel).



Example:

PLOT,0,0

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

4.6.39 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 -- for dramatic effect.

4.6.40 Right (R)

Syntax:

R,num

where:

num is a number (0 - 65535)

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.

4.6.41 Reset Attributes (RE)

Syntax:

RE,attribute,attribute,...
RE,ALL

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

Example:

RE,QS,BL

This will turn Quad-size and blinking to OFF.

4800-Ex Manual
January, 1990

4.6.42 Restore Attributes (RSTA)

Syntax:

RSTA,reg

where:

reg is a register number (1 - 500)

Restores the attribute values saved in the specified register by an earlier Save Attributes (SAVA) command. See Set Attributes Command (SE).

4.6.43 Save Attributes (SAVA)

Syntax:

SAVA,reg

where:

reg is a register number (1 - 500)

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).

4.6.44 Set Background Color (SBC) (Color Terminal Only)

Syntax:

SBC,color

where:

color is any of the following values:

BLK -- black	0 -- black
BLU -- blue	1 -- blue
GRN -- green	2 -- green
CYN -- cyan (blue-green)	3 -- cyan
RED -- red	4 -- red
MAG -- magenta (purple)	5 -- magenta
YEL -- yellow	6 -- yellow
WHT -- white	7 -- white

This command sets the background colors of a color terminal.

Example:

```
SFC,RED: SBC,6  
SE,QS,BL  
"OVERLOAD"
```

This sets red letters on yellow background, then the letters to blinking and quad-size. It then displays the message with those attributes.

4.6.45 Set Attributes (SE)

Syntax:

SE,attribute,attribute,...

where:

attribute 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 (not available on color terminals)
HI	High Intensity characters (not available on color or flat panel terminals)
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. The available attributes are:

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 below:

QS, DS, DH, DW - Refer to the terminal 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 terminal 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 terminal 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 Terminal 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 Terminal 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 Terminal 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 Terminal Manual).

Example 1:

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

This 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:

The following program will display the left and right half of a tank bottom by using graphics characters.

```
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"
```

4.6.46 Set Bit (SETB)

Syntax:

SETB,reg,bit

where:

reg is the register in which the bit is located (1 - 500)

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.

4.6.47 Set Foreground Color (SFC)

Syntax:

SFC,color

where:

"color" can have any of these values:

BLK -- black	0 -- black
BLU -- blue	1 -- blue
GRN -- green	2 -- green
CYN -- cyan (blue-green)	3 -- cyan
RED -- red	4 -- red
MAG -- magenta (purple)	5 -- magenta
YEL -- yellow	6 -- yellow
WHT -- white	7 -- white

This command sets the foreground colors of a color terminal.

Example:

SFC,RED: SBC,6

This example sets the foreground color to red, and the background yellow.

4.6.48 Stop Program Execution (STOP)

Syntax:

STOP

The STOP command terminates execution of the currently executing program. No further programs will be executed until the user selects "Execute" from the Program Utilities Menu or from the editor, or until the remote command "Execute Program Block" is received over an input port.

Example:

STOP

4.6.49 Subtract from Register (SUB)

Syntax:

SUB,m,#n

where:

#n is a register and m is a numeric value

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 Reg. #81 from Reg. #100.

Example 2:

SUB,"A",#100

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

4.6.50 Transfer Data (TR)

Syntax:

TR,source,destination

The terminal has several data input and output devices:

Input Devices

KB (keyboard and keypad)
SI (serial input)
#n (data register)
\$n (data register pointer)
Literal (number or string)

Output Devices

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

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

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

4.6.50.1 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)	Four blanks followed by three digits
(dd.dd)	Four digits with decimal point and two blanks
(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.

4.6.50.2 **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

4.6.50.3 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",destination(aaaaa)
```

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

Example 2: Some characters are valid.

```
TR,"A12$a",destination(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,destination(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),destination(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,destination(ccccc)
```

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

Example 2: Some characters are valid.

```
TR,"A12$a",destination(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,destination(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),destination(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",destination(llllll)
```

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

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

TR,#30,destination(II)

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),destination(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 the user wants 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 n 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),destination(IIIIIIIIII)

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),destination(IIIIn)

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.

4.6.50.4 Numeric Data Types

There are three numeric data types:

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

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

TR,#30,destination(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, the user 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),destination(numeric)).
- 6) Different numeric picture elements can be combined in different pictures (e.g.: TR,source(ddddd),destination(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(LLI)), 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 terminal.

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.

4.6.51 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 for monochrome, 0-49 for color and flat panel)

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

4.6.52 Up (U)

Syntax:

U,n

where:

n is a number.

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.

4.6.53 Vertical Bar Up (VB or VBU)

Syntax:

VBU,xstart,ystart,lngth,maxlngth
- or -
VB,xstart,ystart,lngth,maxlngth

where:

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

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

maxlngth the 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 <new height> 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 <max height> is erased. The parameters <new-height> and <max-height> 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 terminal, <new-height> would be 78. The parameter <max-height> specifies how high a bar to erase before the bar of <new height> is drawn. Therefore, <max-height> should be no less than <new-height>.

The maximum height for <new-height> and <max-height> is 255 for monochrome, 250 for color, and 200 for flat panel.

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 column and row coordinates found in registers #40 and #41 respectively, height of bar is 20 characters.

* The character cells are defined as:

Monochrome terminal: each character cell is 5 units wide (1 unit = 1/5 character cell)

Color terminal: each character cell is 8 units wide (1 unit = 1/8 character cell)

Flat Panel terminal: each character cell is 8 units wide (1 unit = 1/8 character cell)

4.6.54 Vertical Bar Down (VBD)

Syntax:

VBD,xstart,ystart,lngth,maxlngth

where:

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

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

maxlngth is the 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.

* The character cells are defined as:

Monochrome terminal: each character cell is 5 units wide (1 unit = 1/5 character cell)

Color terminal: each character cell is 8 units wide (1 unit = 1/8 character cell)

Flat Panel terminal: each character cell is 8 units wide (1 unit = 1/8 character cell)

4.6.55 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 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.

4.6.56 XOR

Syntax:

XOR,#n,m

where:

#n is a register and 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 terminal with the 4800-E10 option installed. To create this display, it is not required that the terminal be connected to a host or a secondary device. When the terminal 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 terminal if the remote keyboard is attached to the terminal. Programs cannot be entered on the terminal keypad. Remote keyboards are available separately from XYCOM as option 4810-KYB or 4800-K1.

There are a couple of terms which will make it easier to understand what is happening when typing a program into the terminal. A screen display is generated whenever a program stored in a program block is executed. There are 255 program blocks on the terminal, and each of them can store a separate program. When the 4800-E10 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 XYCOM's easy-to-use OIL. Each command in OIL will give the terminal 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 terminal'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 terminal.

This sample program (Figure 4-2) illustrates how easy it is to create a screen display on the terminal. However, it reveals only a tiny fraction of all the capabilities of the terminal. This program is ready to be typed into a program block (see Chapter 3 for use of the terminal's editor).

Screen Display

The screen display which the sample program will create looks like this (see Figure 4-2):

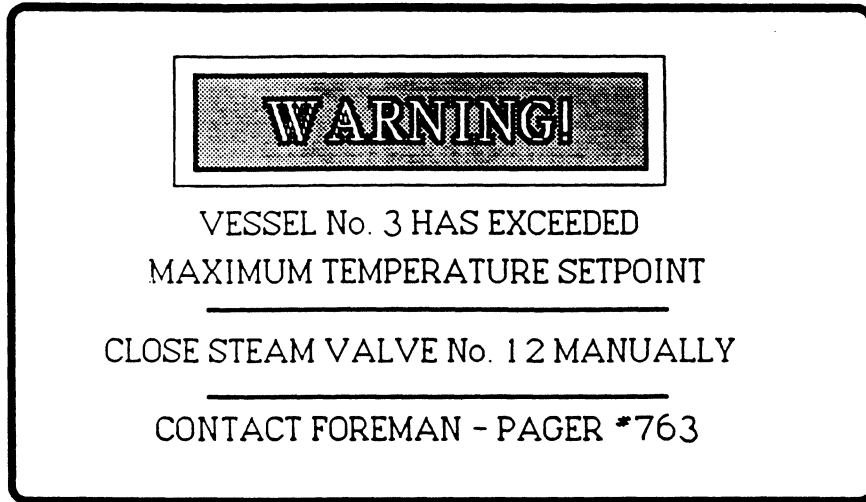


Figure 4-2 Screen Created By Sample Program

```
;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
```

In any line, everything following a semicolon is regarded by the terminal 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.

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 terminal 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 <RETURN>. Program block #1 (the sample program) will be executed.

Chapter 5 PROGRAMMING

5.1 INTRODUCTION

While Chapter 4 described the syntax of all the available OIL commands, this chapter provides more detailed examples of how the OIL commands can be used together in typical terminal 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

Also, some guidelines are given for optimizing the speed of OIL programs.

5.2 DATA REGISTERS

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

These registers are no different than the 500 standard data registers except that they will always reside in CMOS RAM. The terminal 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 #10 are special-purpose registers. Registers #1 through #7 contain the time and date, and are automatically updated by the terminal. Registers #8 through #10 hold the number of characters currently in the keyboard queue and serial port input queue, respectively. Registers #1 through #10 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 -- Day	(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 #11 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 terminal provides two queues, the Keyboard Input Queue and the Serial Port Queue. While the terminal is executing a program block, it is constantly monitoring the keyboard, the keypad, 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. Everything that is received over the serial port is first examined by the terminal 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 terminal 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, the user 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).

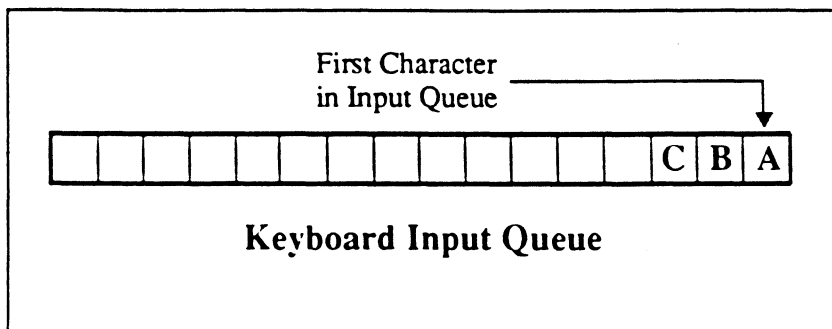


Figure 5-1 Keyboard Input Queue Example (1)

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:

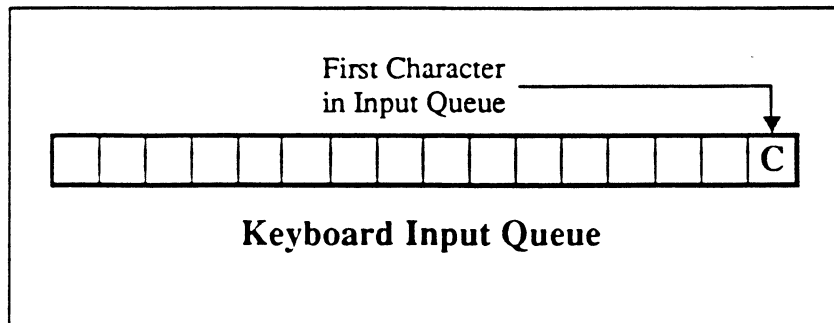


Figure 5-2 Keyboard Input Queue Example (2)

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 and #9

Queue Status Register	Associated Queue
#8	Keyboard Input Queue
#9	Serial Data 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
```

The user 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 the user 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 serial port, displays it, and writes the temperature to the control system. In addition, if the temperature exceeds 500, the terminal will execute program 100, which displays a warning, then transfers the character "T" and temperature to the control system via the serial port.

Program Block 90

```
;Read temp from SI, display, and write to SO
CS:RE CU
%LOOP
@10,10
SE DS
"TEMPERATURE: "
SE BL
IF #45="F":GO END
TR SI(dddd),#50
TR,#50,SC(dddd)
IF #50>500:ES#100
GO LOOP
%END
```

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.

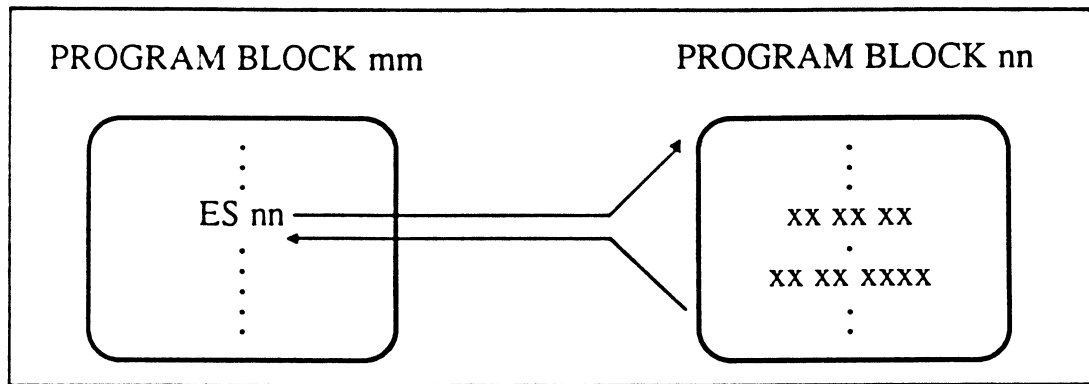


Figure 5-3 Nested Blocks

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, the user 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"

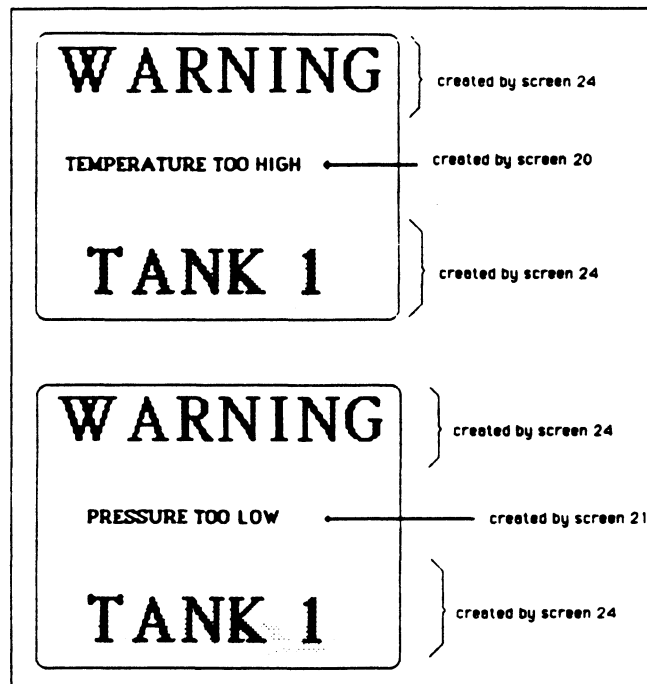


Figure 5-4 Nested Screen Example

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 the user should execute any Set Attribute command before starting the loop rather than within it.

The terminal 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
```


Chapter 6

REMOTE COMMANDS

6.1 INTRODUCTION

A host computer can control the display on the 4800-series terminal by sending a small set of commands, called remote commands, over the terminal 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

These few commands are all that are needed to make full use of your intelligent XYCOM terminal. Because the terminal handles complex operations by using OIL commands, the interface protocol to your control system can be kept simple.

Note, however, that a host computer cannot send OIL commands to be executed directly by the terminal. OIL commands can be entered into a program block either by typing them at the terminal or by downloading them over the serial port (see Appendix A). 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 terminal 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. (The lone exception is the remote command Reset, which will reset the terminal immediately.)

A host computer can control terminal 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 terminal Data Registers. The terminal 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 terminal serial input port. The terminal 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>;...<#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>...<7F>

where:

<block#> is 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#> is a string of up to three ASCII decimal digits specifying a data register in the range 1 through 500 (e.g., "25" for register #25).

<data>	is 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	consists of two bytes (MSB,LSB) specifying the value to be written in data register #n (max. value 65635).
	is the ASCII delete character (7FH).
<text>	is the text of the program block.
REG	is the beginning register number.
NUM	is the number of registers in ASCII decimal.
MSB	is a binary byte.
LSB	is a binary byte.

6.2.1 Reset

Description: The Reset command causes the terminal to initialize itself. If the data registers are selected to be in volatile RAM, they are set to zero. All data input and output queues are cleared. Finally, 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 terminal to terminate any program currently executing, and to begin execution of 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;nnnnp

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 500 (e.g., "40" for data register #40).
nnnn = 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 terminal 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 500 (e.g., "30" for data register #30).

p = 70H

Returns:

In response to this command, the terminal 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 terminal to set the time-of-day clock. (Note that the clock can also be set from the keyboard through the 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.
; = 3BH
MM = two ASCII digits specifying the month (01 through 12).
; = 3BH
DD = two ASCII digits specifying the day of the month (01 through 31).
; = 3BH
hh = two ASCII digits specifying the hour (00 through 23).
; = 3BH
mm = two ASCII digits specifying the minute (00 through 59).
; = 3BH
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 data registers, beginning with data register #11.

Syntax:
<ESC>[14;000p<reg #11 value><reg #12 value>...

where: <ESC> = 1BH
 [= 5BH
 ; = 3BH
 1 = 31H
 4 = 34H
 ; = 3BH
 p = 70H
 <reg #n value> = <low-byte><high-byte>
 = 7FH

The command must contain all 490 register values (from #11 through #500).
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 terminal 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 terminal.

6.2.8 Execute Subprogram Block

Description This command causes the terminal to suspend any program currently executing, execute the specified program block, and resume execution where it was suspended.

Syntax:

`<ESC>[33;dddp`

where: `<ESC>` = 1BH

`[` = 5BH

`3` = 33H

`3` = 33H

`;` = 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

6.2.9 Receive Program Block

Description This command causes the terminal to accept and store a screen program which is imbedded within the command. The screen must be terminated by a DEL character (7FH).

Syntax:
<ESC>[14;ddd<text of stored screen>

where: <ESC> = 1BH
 [= 5BH
 1 = 31H
 4 = 34H
 ; = 3BH
 ddd = 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

6.2.10 Send a Number of Registers

Description If this command is sent to a terminal, it will cause a "Receive a Number of Registers" command to be generated.

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 in ASCII decimal representation

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 character.

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>...<7FH>

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

6.3 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
```

Chapter 7

4800-E1 SERIAL EXPANSION MODULE

7.1 INTRODUCTION

The 4800-E1 Expansion Module adds screen memory, TOD clock, and an RS-232 serial port to the terminal. OIL is not an option on the 4800-E1.

7.2 CONFIGURATION MENU

The Main Menu is modified with the installation of the 4800-E1 Expansion Module. The new menu is shown below.

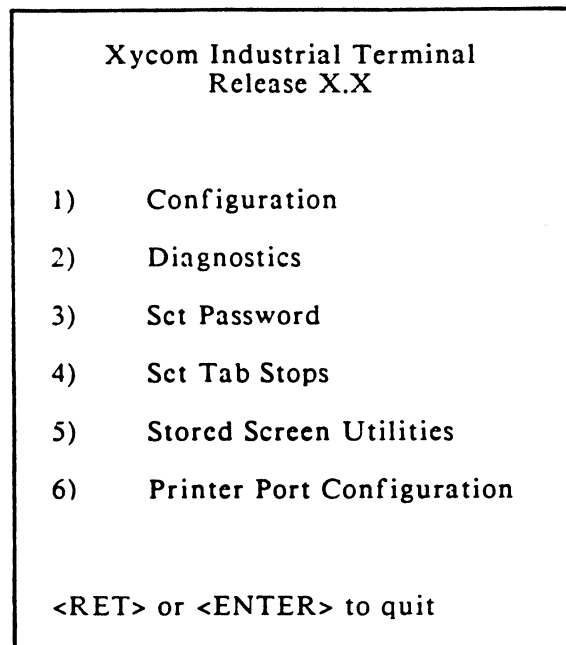


Figure 7-1 Main Menu

If the terminal does not reflect the difference in Main Menus, go back and check the installation procedure steps in Sections 2.2 and 2.3.

The configuration menus are called up from the Main Menu (discussed in Chapter 3). The Printer Port Menu looks like the following:

```

-- Printer Configuration --
6 Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600 7=19.2K
0 1=Parity Enabled          0=Disabled
0 1=Even Parity            0=Odd
0 1=8 Data Bits            0=7 Data Bit

Use <UP ARROW>, <DOWN ARROW>, <LEFT ARROW>, <RIGHT ARROW>.
Use values 0 through 9. <RET> or <ENTER> to quit.
```

Figure 7-2 Configuration Menu

Baud. The baud rate of the channel should be set to match that of the peripheral device.

Parity. There are two parity options. The first will enable or disable the parity test. The second will set the parity type used in communication on the line to either odd or even. The type selected should match the other communication device(s).

Data Bits. This sets how many Data Bits will be used in communications with the peripheral device to either eight or seven. The number selected should match the other communication device.

7.3 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plugs.

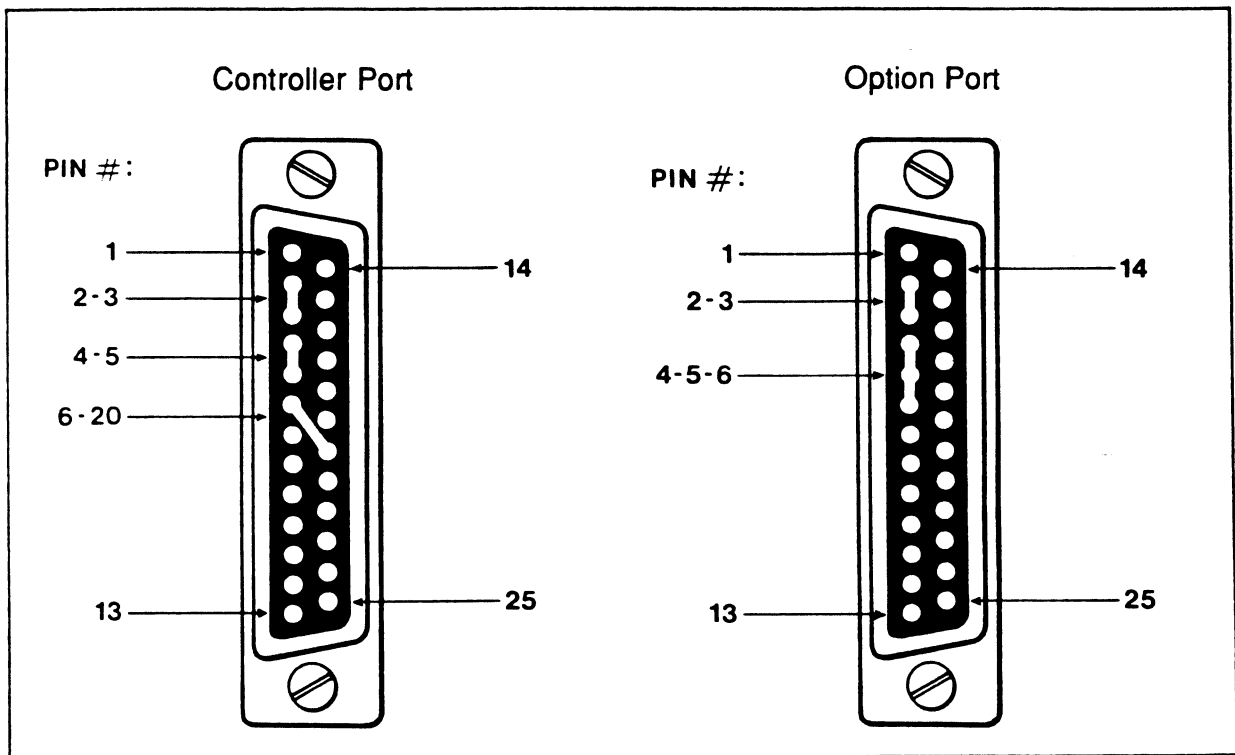


Figure 7-3 Serial Port Test Plugs

7.4 PRINTER PORT COMMANDS

The printer port is enabled or disabled by issuing one of two "remote commands" to the terminal (refer to your terminal manual for information on using remote commands and the command format). The printer port remote commands are as follows:

7.4.1 Enable Printer Port

Function: Enables the printer port, and causes any text sent to the terminal to be echoed to the printer port as well.

Hazeltine emulation: 7EH 2AH

ANSI emulation: <ESC>[=5h

7.4.2 Disable Printer Port

Function: Disables the printer port.

Hazeltine emulation: 7EH 2BH

ANSI emulation: <ESC>[=5l (where the last character is a lower case "L")

The printer port output buffer is 64 bytes long. If the character rate of the printer is slower than the character rate of the terminal, the terminal must wait each time it has a character to send to the printer. While it is waiting, the terminal can not display any other characters or execute any commands coming in over the serial port. Thus, if handshaking is not used, the terminal's input buffer could fill up and data would be lost.

The printer enable/disable commands can be used in conjunction with two other remote commands for enabling/disabling the terminal screen to provide a variety of terminal screen/printer port combinations. Thus, these remote commands could be used to cause the text to appear on both the screen and the printer port, or on just the screen, or on just the printer port.

7.4.3 Enable Screen

Function: Enables the screen display.

Note The screen display is enabled on power-up.

Hazeltine emulation: 7EH 28H

ANSI emulation: <ESC>[= 4h

where: ESC = 1BH

7.4.4 Disable Screen

Function: Disables the screen display. Text destined for the screen will not be displayed.

Hazeltine emulation: 7EH 29H

ANSI emulation: <ESC>[= 4l (where the last character is a lower-case "L")

where: ESC = 1BH

When the screen is enabled all remote commands are processed normally. When the screen is disabled all remote commands (except those to enable/disable the printer or screen) and all text will be ignored and will not affect the screen display.

When the screen and printer port are both enabled most control characters and text are sent to both the screen and the printer. Remote commands go only to the screen and are not sent to the printer. If sequences of control characters (which must be sent to the printer) contain any of the lead-in control characters for valid ANSI or Hazeltine commands (i.e., Tilde or "ESC" followed by valid command character), then the first one or two characters will be treated as remote commands and will not go to the printer. If the characters immediately following these first characters do not constitute a command, they will be passed "as is" to the printer port.

When the screen is disabled and the printer port is enabled, all control codes and text will be sent to the printer except for sequences that look like the remote commands to enable/disable the screen or the printer port.

7.5 REMOTE COMMANDS

There are no special PLC-associated commands in the 4800-E1. None of the OIL commands apply to this Expansion Card.

The following are descriptions of the additional remote commands made available with the installation of the 4800-E1 Expansion Module:

7.5.1 Execute Stored Screen

Function: Causes the terminal to execute the specified screen program. If this command is included in a screen program, after the specified program is executed, control is returned to the calling program. Screen program nesting can be up to 10 levels deep.

Hazeltine emulation: 7EH 10H <P1>

ANSI emulation: <ESC>[12;<P1>p

where: <ESC> = 1BH
<P1> = Screen number (1-255)

7.5.2 Receive Stored Screen

Function: Causes the terminal to accept and store a stored screen which must be imbedded within the command. This screen must be terminated by DEL (7FH).

Hazeltine emulation: 7EH 1EH <P1>

ANSI emulation: <ESC>[14;<P1>p

where: <ESC> = 1BH
<P1> = Screen number (1-255)

Example: To send screen 5:

In Hazeltine mode:

Host sends: 7EH 1EH 05H <text of stored screen #5> 7FH

In ANSI mode:

Host sends: <ESC>[14;5p <text of stored screen #5>

7.5.3 Transmit Stored Screen

Function: Causes the terminal to transmit a Receive Stored Screen command, followed by the actual stored screen, followed by DEL (7FH).

Hazeltine emulation: 7EH 1BH <P1>

ANSI emulation: <ESC>[13;<P1>p

where: <ESC> = 1BH
<P1> = Screen number (1-255)

Example: To send screen 35:

In Hazeltine mode:

Host sends: 7EH 1BH 23H
Terminal responds: 7EH 1EH 23H <text for stored Screen #35>

In ANSI mode:

Host sends: <ESC>[13;35p
Terminal responds: <ESC>[14;35p <text for stored Screen #35>

7.5.4 Jump to Stored Screen

Function: Jumps to the specified stored screen and begins executing that screen. Unlike the Execute Stored Screen command, this command does not return to the calling screen.

Hazeltine emulation: 7EH 24H <P1>

ANSI emulation: <ESC>[19;<P1>p

where: <ESC> = 1BH
<P1> = Screen number (1-255)

7.5.5 Copy Screen Program

Function: Copies the contents of any screen program into another screen program.

Hazeltine emulation: 7EH 2CH <P1><P2>

ANSI emulation: <ESC>[23;<P1>;<P2>;p

where: <ESC> = 1BH
<P1> = Source screen program
<P2> = Destination screen program

7.5.6 Set Programmable Key

Function: Associates a screen program with a key on the keypad and the 10-key function keypad. Thereafter each time that key is pressed, the contents of that screen program are put into the keyboard buffer. Characters within double quotes are sent as is, all other characters must be pairs of hex digits 0-9/A-F representing byte values. In this way you can set your own "Macro" key definitions.

Hazeltine emulation: 7EH 2DH <K><P1>

ANSI emulation: <ESC>[24;<K>;<P1>p

where: <K> = Key number (0-37, see Table 7-1)
 <P1> = Screen number (1-255)
 <P1> = 0 to reset the key to return to its default value

When in half-duplex mode:

1. Contents are put in the output buffer.
2. Contents are displayed on the screen.

When the terminal is in multi-drop mode and a keypad or 10-key function keypad key associated with a program is pressed, the following happens:

1. The contents of the program are transmitted out the serial port.
2. The program is executed (treated as if it were received from a host) by the terminal and the results are displayed on the screen.

Table 7-1 Keypad Key Numbering

NO.	KEY	NO.	KEY	NO.	KEY
0	A	13	F2	26	ENTER
1	B	14	up arrow	27	F4
2	C	15	down arrow	28	PF1
3	D	16	8	29	PF2
4	E	17	5	30	PF3
5	F	18	2	31	PF4
6	F1	19	.	32	PF5
7	F6	20	F3	33	PF6
8	left arrow	21	F5	34	PF7
9	7	22	right arrow	35	PF8
10	4	23	9	36	PF9
11	1	24	6	37	PF10
12	0	25	3		

7.5.7 Set Executable Key

Function: This command allows a keypad or program function keypad key to be associated with a screen program. When the key is pressed the associated screen program is executed.

Hazeltine sequence: 7E 39 <k> <P1>

ANSI sequence: ESC [29; <k>; <P1> p

where:

<P1> = stored screen number 1 - 255 (0 to disable executable key)

<k> = keypad or program function keypad key number (0-27, see Table 7-1)

This command differs from the **Set Programmable Key** command in that the screen program is actually executed. The **Set Programmable Key** command is used to redefine the code sent by the terminal when a key is pressed.

The screen number associated with the **Executable Keys** is stored in CMOS RAM. Once the key is set, it remains set until either the CMOS RAM is cleared by disconnecting the battery or the **Set Executable Key** command is issued for the key with a screen number of 0.

The executable keys function only when the terminal is in operating mode. They do not function when in the stored screen utilities or any of the other menus.

If any other screen programs are executing (perhaps as a result of receiving an **Executable Screen Program** remote command) when an "executable" keypad key is pressed, the key's screen program is not executed until the other screen programs finish. (An Executable Key will therefore not interrupt "infinite loop" screen programs which re-execute themselves.) After completing a remote command, but before executing any further remote commands, the terminal will first execute the screen programs of any Executable Keys that have been pressed.

7.5.8 Enable Output Echo

Function: This command allows data in a screen program to be put in the multidrop output buffer from within a screen program.

Hazeltine sequence: 7E 3A

ANSI sequence: ESC [=8h

This command is used with the **Set Executable Key** command. Normally Output Echo is Disabled, but when enabled within a screen program, data from the screen program is put in the output buffer just as if it were typed in from the keyboard. The maximum length of a record is 127 bytes. Anything over 127 bytes will be lost. A carriage return must be supplied to complete the record.

In a typical application the user would use the **Set Executable Key** command to associate a screen program with a keypad key. When the user pressed the key, the screen program would be executed. The screen program would contain data to be displayed on the screen or even a new screen layout. By using the **Enable Output Echo** command in the screen program, data would be put into the multidrop output buffer. This data would be read by the host the next time the terminal was polled with the **Select for Output** command.

Once a record has been completed by sending a carriage return, all subsequent characters will be ignored until data has been removed by the host through a **Select for Output** command.

This allows the terminal's display to be updated immediately when a keypad key is pressed rather than waiting for the host to see that a key has been pressed when it does a **Select for Output** and then have to immediately do a **Select for Input** and then send an **Execute Stored Screen** command to the terminal. The visual response time to an operator's key press is then not dependent on the polling time.

The **Output Echo** command should only be enabled within a screen program. The screen program should bracket the data to be put in the output buffer with the **Enable Output Echo** and **Disable Output Echo** commands. The Screen Output should also be disabled if the user wants to put control characters in the output buffer. If the screen is not disabled, only ASCII characters will be put in the output buffer.

The **Enable Screen Output** command should be issued at the same time **Output Echo** is disabled. When screen output is disabled, most commands will be ignored. The descriptions of the **Enable/Disable Screen Output** and **Enable/Disable Printer Output** commands contain more information on the effects of disabling screen output.

In multidrop mode the output buffer is "complete" when a carriage return is put into it or Keyboard Character Mode is selected and a character put into it. When the output buffer is full, the keyboards and keypads are locked so typing an "Executable" keypad key will have no effect. The buffer is made empty when the terminal is polled by the host with the **Select for Output** command.

Data put in the output buffer without making it "complete" is not sent to the host when polled with the **Select for Output** command.

Once the output buffer is made full with data in a screen program using the **Enable Output Echo** command, any other data the program tries to put in will be lost.

7.5.9 Disable Output Echo

Function: This command disables the output echo described above.

Hazeltine sequence: 7E 3B

ANSI sequence: ESC [=8l

where:

l = lower case "L"

7.6 STORED SCREEN UTILITIES MENU

The stored screen utilities menu is invoked from the main menu. The following menu options are provided:

- Edit** Enter a new screen or edit an existing screen. You will be prompted for the number of the screen to edit (1-255).
- Execute** Prompts the user for the number of the screen to be executed.
- Copy** One stored screen can be copied to another. You will be prompted for the source screen (to be copied) and the destination screen (into which the source screen will be copied). All data currently in the destination screen will be lost, replaced entirely by the data in the source screen. Copying from an empty screen is also a convenient way of emptying out the destination screen.
- Transmit** To another terminal or system. The user is prompted for the number of the screen to transmit (or all screens), and the number of the destination screen.

NOTE

If you choose to transmit all the screens, all 255 screens, including empty screens, will be transmitted. In particular, if empty screens are transmitted to another terminal they will erase screens of the same number already stored on the receiving terminal (see Appendix A).

- Directory** Displays the first line of each stored screen.
- Verify** Compares a downloaded image with the contents of a specified stored screen or all stored screens (refer to Appendix A). This option is useful if option 4 (Transmit Screen) was used to backup screen programs to a tape or to another computer system. Refer to Appendix A for instructions about storing screen programs on tape.
- Receive** Allows the terminal to receive screens from another terminal or system.

NOTE

Screens are transmitted, received, and verified over the printer port when in the "Stored Screen Utilities".

Two versions of each screen are included: the Hazeltine 1500 and the ANSI versions. (Each remote command exists in two forms, depending upon whether Hazeltine 1500 or ANSI emulation is desired.) To execute a stored screen written in Hazeltine 1500 format, the terminal must be configured for Hazeltine 1500 emulation. Likewise, to execute a screen written in ANSI format, the terminal must be configured for ANSI emulation.

Monochrome Screen Program Example

Hazeltine 1500

```
7E 09 01 12 05 36 0A;Draw Box command<CR>
7E 11 13 06          ;Cursor to X,Y command to put cursor in box<CR>
7E 03 09 02          ;Change Char. Attributes command to quad,<CR>
                    ;blink, reverse<CR>
"WARNING"            ;message in box<CR>
```

ANSI

```
"<ESC>[2;1;6;19;11;55p" ;Draw Box command<CR>
"<ESC>[7;20H"           ;Cursor to X,Y command to put cursor in box<CR>
"<ESC>[1;9;2p"          ;Change Char. Attributes command<CR>
"WARNING"               ;message in box<CR>
```

Color Screen Program Example

Hazeltine 1500

```
7E 03 40 10          ;Select red on blue characters<CR>
7E 09 01 12 05 36 0B;Draw a box<CR>
7E 11 13 06          ;Position cursor inside of box<CR>
7E 03 48 12          ;Select quad blinking, red on blue<CR>
"WARNING"<CR>
```

ANSI

```
"<ESC>[1;64;16p"      ;Select red on blue characters<CR>
"<ESC>[2;1;6;19;12;55p" ;Draw a box<CR>
"<ESC>[7;20H"         ;Position cursor inside of box<CR>
"<ESC>[1;72;18p"     ;Select quad blinking, red on blue<CR>
"WARNING"<CR>
```

If displayable characters or remote commands are encoded in ASCII, the sequence of ASCII characters must be enclosed in quotation marks ("). This is because all characters are interpreted as hex, unless enclosed in quotation marks. It is recommended that each separate line be enclosed in quotes, as in the above example. If the above example were enclosed in only two quotes, one at the beginning and one at the end, the <CR>s at the end of each line would be interpreted as remote commands and executed and the comments would be printed, thereby affecting the current cursor position. If each line is embedded in quotes, the <CR> at the end of each line is not interpreted as a remote command.

Note also that it is not required that Hazeltine-format screens be entered in hex and ANSI-format screens in ASCII. Hazeltine-format commands may also be entered in ASCII (if the commands are enclosed in quote marks), and ANSI-format commands in hex. However, since ANSI emulation is designed so that ASCII characters can be used, ANSI-format commands will typically be entered in ASCII.

Likewise, Hazeltine-format commands will typically be entered in hex. Note that commands entered as ASCII characters generally take up less program storage than hex codes, since it takes two hex digits to reproduce a single ASCII character.

The <CR> and <ESC> shown above actually take up only one character location apiece on the terminal screen. <CR> is visible so the user can see the end of the line.

Characters representing hex bytes (i.e., 7E 01 07) must always appear in pairs. For example, 1 is not a legal representation of the hex byte 01. The leading zero is required.

Spaces not enclosed in quotes are ignored. For example, either 7E 01 3E or 7E013E is an acceptable way of representing 3 hex bytes.

Any text appearing between quotes (") is stored as ASCII characters, and is used literally. All other text represents hexadecimal bytes. A double quote (") that appears between quotes will cause a single quote to be displayed when the screen is executed.

Any text following a semicolon (;) outside of quotes is a comment. Comments extend to the end of the line. When a stored screen is executed, comments are ignored.

It is a good idea to include an identifying comment as the first line of each stored screen so that it will show up in the directory entry for the screen.

7.8 NESTED SCREENS

The remote command `Execute Stored Screen` can be included within a screen program. For example, suppose screen `mm` contains an `Execute Stored Screen nn` command. When Screen `mm` is executed, all the remote commands will be executed in the order in which they occur. When the `Execute Stored Screen nn` command is executed, control will be transferred to Screen `nn`, which will be executed in its entirety. When Screen `nn` is finished, control will be returned to Screen `mm`, and execution of subsequent commands in screen `mm` will continue.

Any number of `Execute Stored Screen` commands can be included in one screen, and they may be included anywhere within the screen. Up to ten levels of nesting is permitted.

Stored Screens 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 screen, while each of the possible variations may be placed in separate screens. These variations can be called from other screens as needed. Then instead of redesigning each screen from scratch, the user may design a screen in parts and reuse these parts in other screens.

Two simple examples are provided below (one for Monochrome terminals and one for Color terminals). Suppose an application is monitoring a vat for two danger 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 letters "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 screens (see Figure 7-4).

- Screen 24: Creates the display "WARNING"
- Screen 20: Creates the display "TEMPERATURE TOO HIGH"
- Screen 21: Creates the display "PRESSURE TOO LOW"

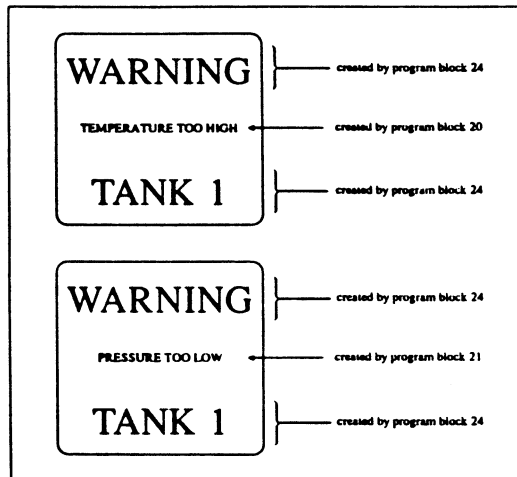


Figure 7-4 Nested Screen Example

To display the "TEMPERATURE TOO HIGH" warning, screen 7 would be executed.

To display the "PRESSURE TOO LOW" warning, screen 8 would be executed. Screen 7 contains an Execute Stored Screen command to execute Screens 24 and 20, while screen 8 executes Screens 24 and 21.

Monochrome Example

Screen 7 (For "TEMPERATURE TOO HIGH" Warning on Tank 1)

```
7E 10 18      ;Execute Stored Screen 18H (=24)
7E 10 14      ;Execute Stored Screen 14H (=20)
```

Screen 8 (For "PRESSURE TOO LOW" Warning on Tank 1)

```
7E 10 18      ;Execute Stored Screen 18H (=24)
7E 10 15      ;Execute Stored Screen 15H (=21)
```

Screen 24

```
7E 11 1D 03   ;Cursor to X,Y command
7E 03 1A 01   ;Blinking, double-size, high-intensity
"WARNING"
7E 11 1E 11   ;Cursor to X,Y command
7E 03 02 01   ;Double-size, high-intensity
"TANK 1"
7E 11 1B 0B   ;Cursor to X,Y command
7E 03 00 00   ;Default Character Attributes
```

Screen 20

```
"TEMPERATURE TOO HIGH"
```

Screen 21

```
"PRESSURE TOO LOW"
```

Color Example

Screen 7 (For "TEMPERATURE TOO HIGH" Warning on Tank 1)

```
7E 10 18 ;Execute Stored Screen 18H (=24)
7E 10 14 ;Execute Stored Screen 14H (=20)
```

Screen 8 (For "PRESSURE TOO LOW" Warning on Tank 1)

```
7E 10 18 ;Execute Stored Screen 18H (=24)
7E 10 15 ;Execute Stored Screen 15H (=21)
```

Screen 24

```
7E 11 1D 03 ;Cursor to X,Y command
7E 03 7A 11 ;Double-size, blinking, white on blue
"WARNING"
7E 11 1E 11 ;Cursor to X,Y command
7E 03 72 11 ;Double-size, blinking, white on blue
"TANK 1"
7E 11 1B 0B ;Cursor to X,Y command
7E 03 70 10 ;Regular size, white on blue
```

Screen 20

"TEMPERATURE TOO HIGH"

Screen 21

"PRESSURE TOO LOW"

Using the Serial Line for Input During Screen Program Execution

During execution of a screen program, the terminal monitors the RS-485 serial line, storing any characters received in a serial data buffer. When the last character in the screen program has been read, the terminal proceeds to read and execute the serial data buffer. The sample screens below use this feature to allow the host to select the next screen to be executed:

Monochrome Example

Screen 1

7E 1C	;Clear Screen
7E 11 1D 03	;Cursor to X,Y
7E 03 1A 01	;Change Character Attributes
"WARNING"	
7E 11 1E 11	;Cursor to X,Y
7E 03 12 01	;Change Character Attributes
"TANK 1"	
7E 11 1B 09	;Cursor to X,Y
7E 03 00 00	;Change Character Attributes
7E 10	;First part of Execute Screen command, ;waiting for last byte from host system

Screen 2

"TEMPERATURE TOO HIGH"

Screen 3

"PRESSURE TOO LOW"

Whatever character is received by the terminal over the serial line will determine the next screen to be executed. For example, if 02H is received, Screen 2 will be executed.

Color Example

Screen 1

```
7E 1C ;Clear Screen
7E 11 10 03 ;Cursor to X,Y
7E 03 7A 11 ;Double-size, blinking, white on blue
"WARNING"
7E 11 1E 11 ;Cursor to X,Y
7E 03 72 11 ;Double-size, white on blue
"TANK 1"
7E 11 1B 09 ;Cursor to X,Y
7E 03 70 10 ;Regular size, white on blue
7E 10 ;First part of Execute Screen command,
;waiting for last byte from host system
```

Screen 2

```
"TEMPERATURE TOO HIGH"
```

Screen 3

```
"PRESSURE TOO LOW"
```

Entering Variable Data into a Stored Screen

Nested screens can be used to display changing data in a screen display. For example, suppose that the message to be displayed on the screen is "Temperature of Tank 2 is xxx Degrees", where xxx varies. The following screens could display the message:

Screen 10

```
7E 1C ;Clear Screen
"Temperature of Tank 2 is "
7E 10 0B ;Execute Screen 11
" Degrees"
```

Screen 11

```
Contains a number written by the host
```

The host would periodically write new data to screen 11 by issuing a Receive Screen remote command. For example, to transmit the number "120" to screen 11, the host would transmit the following hexadecimal data to the terminal:

```
7E 1E 0B 22 31 32 30 22 7F
```

Note that 22 is the ASCII value of the quotation mark (").

Chapter 8

4800-E2 MULTI-DROP EXPANSION MODULE

8.1 INTRODUCTION

The 4800-E2 Expansion Module adds screen memory, TOD clock, and two RS-485 serial ports to the terminal. OIL is not an option on the 4800-E2.

The RS-485 ports are designed to be used to network a number of Xycom terminals in a multidrop network. This network is described later.

8.2 CONFIGURATION MENU

The Main Menu is modified with the installation of the 4800-E2 Expansion Module. The new menu is shown below.

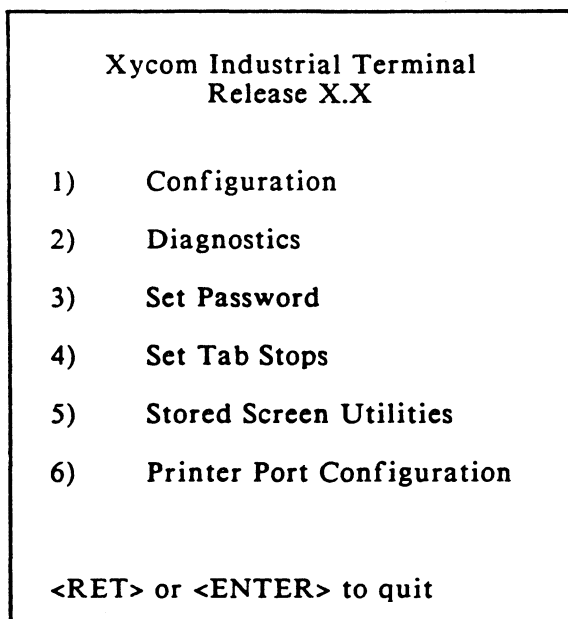


Figure 8-1 Main Menu

Items #5 and #6 are new. If the terminal does not reflect the difference in Main Menus, go back and check the installation procedure steps in Sections 2.2 and 2.3.

The Configuration Menu is the same as the standard (unexpanded) terminal Configuration Menu, with two additions pertaining to multidrop operation. These are discussed below.

Terminal Address. This sets the address of this terminal in the range 1 to 63 (decimal) for input or output in multidrop mode. An address of 0 disables the terminal from the multidrop network.

Minimum Reply Delay Time. The "Minimum Reply Delay Time" option is used only when the terminal is in multidrop mode. The time specified is in milliseconds and can be a value from 0 to 999. This is a minimum time, depending on what the terminal is doing and what data is already in its receive buffer when a "Select for Output" or "Report" type command is received.

This option specifies the minimum amount of time the terminal will wait before transmitting data to the host after receiving a "Select for Output" command or any other "Report" command which causes the terminal to send data to the host (Return Cursor Position, Transmit Stored Screen, etc.,).

This gives the host adequate time to disable its RS-485 transmitter and enable its receiver so it can receive the data from the terminal.

The configuration menus are called up from the Main Menu (discussed in Chapter 3). The Printer Port Menu looks like the following:

```

-- Printer Configuration --
6 Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600 7=19.2K
0 1=Parity Enabled          0=Disabled
0 1=Even Parity            0=Odd
0 1=8 Data Bits            0=7 Data Bit

Use <UP ARROW>, <DOWN ARROW>, <LEFT ARROW>, <RIGHT ARROW>.
Use values 0 through 9. <RET> or <ENTER> to quit.
```

Figure 8-2 Configuration Menu

Baud. The baud rate of the channel should be set to match that of the peripheral device.

Parity. There are two parity options. The first will enable or disable the parity test. The second will set the parity type used in communication on the line to either odd or even. The type selected should match the other communication device(s).

Data Bits. This sets how many Data Bits will be used in communications with the peripheral device to either eight or seven. The number selected should match the other communication device.

8.3 SERIAL LOOP BACK TEST

The ports on the terminal can be tested from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plug used on the serial port. The test set-up used for the RS-485 ports is shown later.

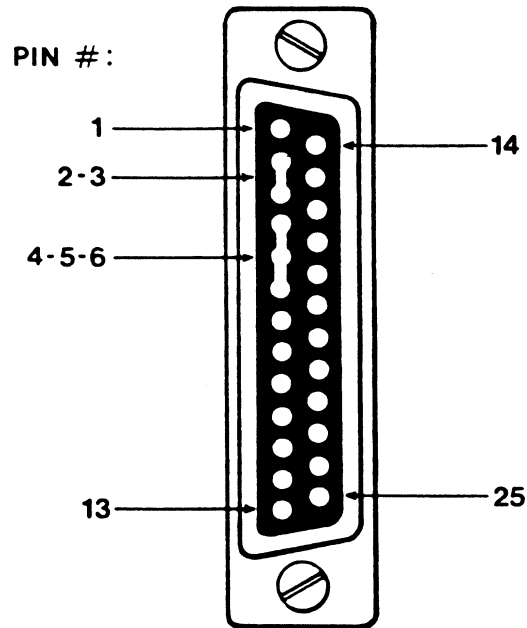


Figure 8-3 Serial Port Test Plug

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

- Controller port time out error.
- Controller port data error.
- Controller port CTS-RTS failure.
- Controller port DTR-DSR failure.

RS-485 Test

Selection #9 of the Diagnostics Menu is a test of the RS-485 multidrop serial port. This test requires an independent 4800-MDA (MultiDrop Adapter) unit. A cable must be run from the 4800-E2 to the Xycom 4800-MDA and back. Figure 8-2 shows Test Cable 1, from terminal RS-485 port A to the 4800-MDA port A. Figure 8-3 shows Test Cable 2, back from the 4800-MDA port B port to the terminal RS-232 port.

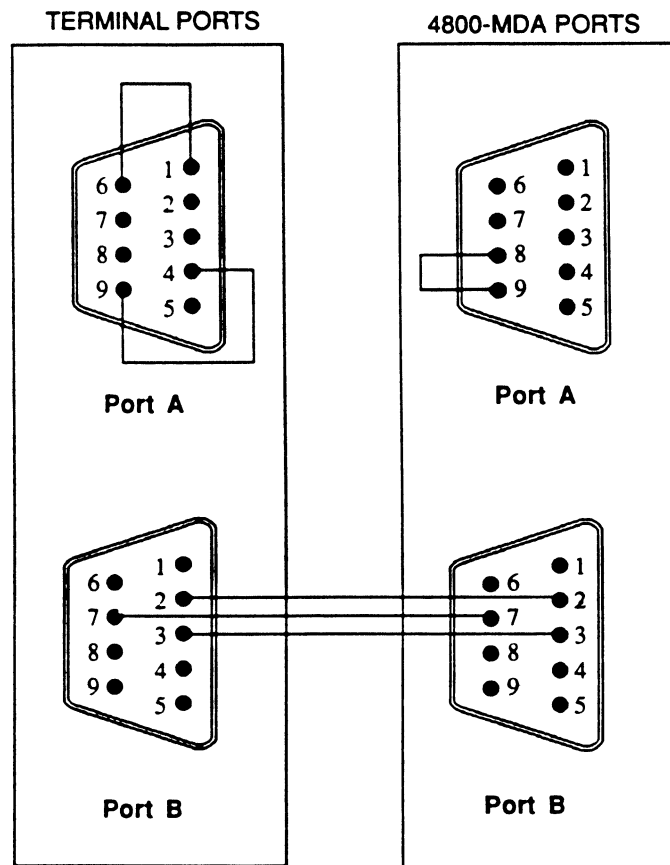


Figure 8-4 Test Cable 1

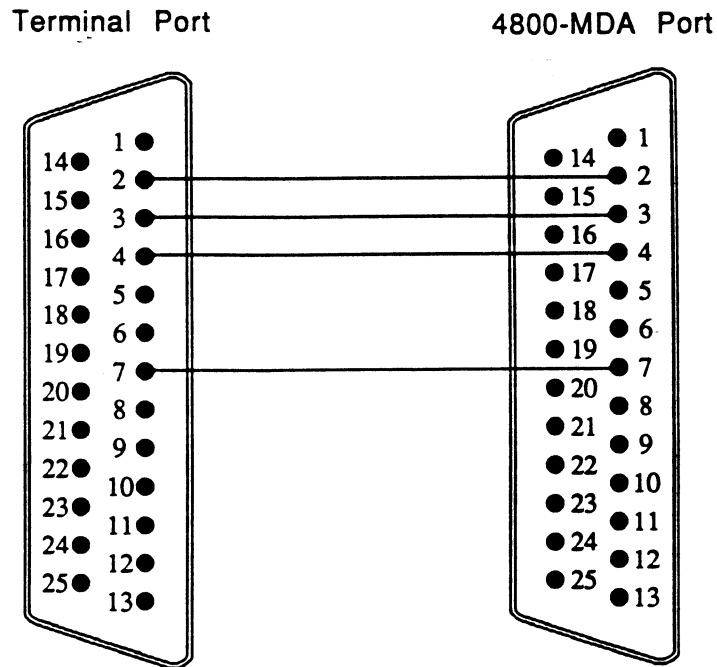


Figure 8-5 Test Cable 2

With the construction and installation of these two Test Cables, the RS-485 diagnostics test can be run by selecting #9 from the diagnostics menu.

If the Diagnostics test runs without any problems, the message "RS-485 Test Passed." will appear on the terminal. The following are valid error messages:

- 4800-E2 UART failed to receive any data.
- 4800-E2 UART received incorrect data.
- Communications Module timeout circuit failed.
- Communications Module timeout circuit failed to reset.
- Controller UART failed to receive any data.
- Controller UART received incorrect data.
- 4800-E2 timeout circuit failed.
- 4800-E2 timeout circuit failed to reset.

8.4 MULTIDROP NETWORK OVERVIEW

The Xycom RS-485, positioned at some point in a network, can communicate with up to 31 terminals. Messages and data are passed from the host computer simultaneously to all of the terminals in the network.

The positions, or terminals, along the network are called nodes. Each node in the network has a unique address (set in the terminal Configuration Menu). Each message or command transmitted by the host is prefaced by the address of the terminal for which it is intended. An addressed terminal can also transmit a message back to the host computer.

The network interconnection can be made from either port on the RS-485, as they are designed in parallel. For correct operation, each end of the network must be terminated. This means that specific resistor values must be connected between the wires of the cables and ground. These terminations are designed into Xycom products; only certain pins need be tied to accomplish the termination.

The Xycom multi-drop network uses a half-duplex "poll/select" protocol. Poll/select refers to the fact that the host computer controls all communications by issuing commands to the terminals. The host can transmit data to any or all of the terminals at one time.

A terminal will transmit data only in response to the host computer. For this to happen, the terminals are polled, one by one or specifically, with a data request from the host. If the terminal has data, it will transmit. If not, it will send a negative reply.

To send data to a terminal, the host will send a **Select for Input** command to a specific terminal. This terminal will accept all data and execute all commands until another command is sent from the host, either to it or another terminal.

To read data, the host computer will issue a **Select for Output** command. The host will periodically send this command to each terminal on the network to check for data.

The host computer must assert an RTS or CTS in order to begin transmission. If more than 37.6 msec elapses between start-bits, the RS-485 interface will return to the receiving state. If this happens, the user must de-assert the signal, then assert it again. This is a fail-safe feature designed into the RS-485 that will not allow more than one transmitter to be active at one time and/or prevent faults at network modes from disabling the system.

8.4.1 Multidrop Commands

To send commands or data to a terminal, the host computer first issues a **Select for Input** command to the terminal. This tells a terminal it should accept all commands and data until another terminal is selected. Then the host computer can send any characters or remote commands (for example, **Draw Box** or **Set Attributes**), which it would normally send to a terminal. All such commands or data will be received and executed by the selected terminal until a **Select for Input** command selects another terminal.

To read data from a specific terminal, the host computer issues a **Select for Output** command to the terminal. In response, the terminal will send one record back to the host computer. A record is a complete line typed on the terminal (0-126 characters followed by the ENTER or RETURN key). If a complete line has been typed on the specified terminal, the terminal will transmit all the characters followed by a Carriage Return and an EOT character. If a complete record (line) is not available, the terminal will transmit a single EOT character. The host computer software will periodically issue **Select for Output** commands to each terminal on the line to check if any has data to transmit.

8.4.2 Operator Input

Keys typed by the user are echoed on the screen and put into the keyboard buffer. The terminal can hold one complete record, i.e. it will wait until it is transmitted then accept any part of another record (the maximum length for a record is 127 bytes). The Enter or Return key terminates the current keyboard buffer record. Before the Enter or Return key is pressed, the operator can use the backspace or left arrow key to delete erroneous keystrokes both from the terminal internal data buffer and from the screen.

Once the Enter or Return is typed, no more keyboard input is accepted until the host reads the keyboard buffer record with the **Select for Output** command. If the terminal receives the **Select for Output** command before the user types Enter or Return, the keyboard buffer record is not sent to the host. Only the EOT (04H) is sent.

8.4.3 Keyboard Echo and Keyboard Character Mode

There are two commands in multidrop mode that allow the Keyboard Echo to be Enabled or Disabled. Keyboard echo enabled is the default state. When Keyboard echo is Disabled in multidrop mode (disable Keyboard echo works only when the terminal is in multidrop mode), keys typed by the user are not echoed on the screen. The keycodes are put into the transmit buffer and can be removed with the backspace key, but nothing is echoed on the screen. If the terminal receives the **Select for Output** command before the user types Enter or Return, the keyboard buffer record is not sent to the host. Only the EOT (04H) is sent.

The Enable Keyboard Character Mode works only when the terminal is in multidrop mode (Keyboard Character Mode Disabled is the default state). When Keyboard Character Mode is Enabled in multidrop mode, a single key typed by the user completes a record. No carriage return is required. Each time the user types a key the terminal will send the keycode (followed by EOT) in response to a **Select for Output** command from the host. In this mode backspace will send the code for a backspace. Multiple keycodes will be sent in the record for those keys that send more than one keycode when typed.

8.4.4 Terminal Addressing and Broadcasting

Each terminal has a unique address assigned by the user from the terminal keyboard through the terminal configuration menu. Valid terminal addresses are 1-31 (base 10).

A host computer can select all terminals for input (a.k.a. broadcasting) by issuing a **Select for Input** command accompanied by address 00. Until another **Select for Input** is issued, all terminals will accept commands from the host. This allows the host to broadcast commands and data to all of the terminals simultaneously.

When selected at the terminal, address 00 disables the multidrop protocol and allows the terminal to function as a non-multidrop terminal (i.e., no select for input or output), using the RS-485 port.

Only the lower 6 bits of the address sent in the **Select for Input** and **Select for Output** commands are used for the address comparison. The upper two bits are ignored. This gets around the problem of trying to send control codes as addresses in Hazeltine mode.

8.4.5 Multidrop System Considerations

When selected for output, a terminal could respond with an EOT (04H) or the first byte of the keyboard buffer record in as little as 60usec.

The response to the **Select for Output** could take longer if the terminal was recently selected for input and it was still executing the commands it received at that time.

In any case, a terminal that is on-line will always respond to the **Select for Output** command. If a terminal does not respond in a fixed amount of time, the host software should assume the terminal is off-line.

The time-out time should be carefully chosen. If the host does not wait long enough before timing-out it may assume an on-line terminal is off-line and start communicating with other terminals on the RS-485 network. When the terminal thought to be off-line finally responds to the **Select for Output** command, it may corrupt the current communications on the network.

The "Minimum Reply Delay Time" option in the Configuration Menu (see Figure 8-2) can be used to specify the minimum time the terminal will wait before transmitting data to the host after receiving a "Report" type command or a "Select for Output" command. This gives the host time to disable its RS-485 transmitter and enable its receiver so it can receive the data from the terminal.

Keys that return control codes will have their codes displayed on the screen using a two-character representation of their ASCII names. The control codes are stored "as is" in the keyboard buffer. If the terminal is in quad-size, double-high, or double-size mode, the control codes will be displayed on the screen as spaces.

8.4.6 Using Remote Commands With a Multidrop Network

As previously stated, any of the remote commands described in your terminal manual can be transmitted to selected terminals in the Multidrop Network. However, there are some remote commands which must be given consideration when programming the host computer.

If a terminal is selected for input, and it receives either the command to Return Cursor Position or the command to Transmit Stored Screen, the terminal will return the requested data to the host computer immediately (i.e. without waiting to be selected for output). This means if the host computer issues either of these commands, it must be ready to receive data from the selected terminal immediately. This situation would also occur if any command to return data to the host were to be executed from within a screen program while a terminal is on the Multidrop Network. Therefore, it is recommended those commands which are designed to return data to the host not be imbedded in any screen programs, unless the host computer is ready to receive data at any time while the screen program is executing.

Likewise, the two multidrop commands (Select for Input and Select for Output) should never be executed from screen programs while a terminal is in multidrop mode. These commands are intended to be issued from a host serial port within the network and they could cause a conflict in the communication protocol.

In addition, while a screen program is executing, a terminal can still receive data and commands. However, commands and data are buffered and are not examined by the terminal until the screen program finishes. Thus, if a terminal is executing a screen program when the host issues it a Select for Output command, the terminal will not respond until it finishes the screen program (by which time the host might have already "timed out"). This situation should be considered when programming the host computer.

8.5 CONNECTING A MULTIDROP NETWORK

Each RS-485 multidrop interface on a Xycom Multidrop Network has 2 ports: port A and port B. The internal configuration of the ports is such that both ports on a terminal are wired in parallel. Therefore, the ports on any given terminal are wired in parallel. Therefore, the ports on any given terminal may be used interchangeably in terms of installing the incoming and outgoing network lines. However, if a terminal is at the head-end or tail-end of a network, the jumper plugs used to facilitate termination can only be installed on port A. Figure 8-6 illustrates a typical hook-up for a 3-terminal multidrop network.

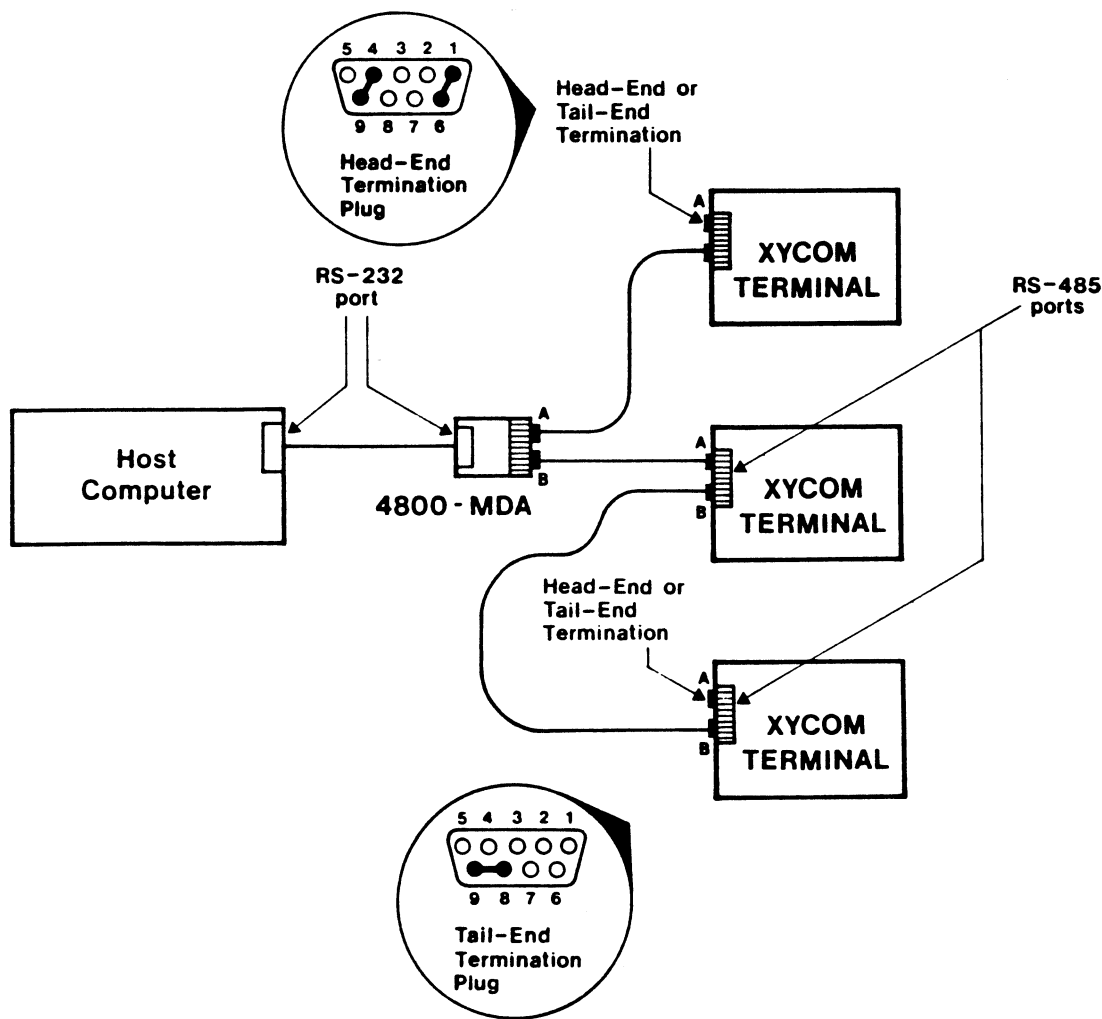


Figure 8-6 Multidrop Connection

Only pins 2, 3, and 7 are used on each port of the RS-485 interface. Note the termination plugs on port A of the 4800-E2 Expansion Module (head-end) and port A of the last terminal in the network (tail-end).

All Xycom RS-485 compatible networking devices are designed with internal termination so they may be used at either the head-end or tail-end of a network. To activate the termination circuitry one need only install jumpers at specific pins on port A of the head-end and tail-end nodes. This is best facilitated by installing a male DB-9 plug on port A of the corresponding device.

8.5.1 Head-end Termination

On port A of the head-end, install a male DB-9 plug with pins 1 & 6 and 4 & 9 shorted together (refer to Figure 8-6). Installing such a jumper plug will terminate the head-end of the network with the proper RS-485 impedance, as well as provide an offset on the line so that during periods of no transmission, all receivers see a marking condition.

8.5.2 Tail-end Termination

On port A of the tail-end, install a male DB-9 plug with pins 8 & 9 shorted together (refer to Figure 8-6). Installing such a plug will terminate the tail-end with the proper RS-485 impedance.

8.5.3 RS-485 Cabling

In situations where cable length is considerable or in electrically noisy environments, a static charge might build up on the shielding of the cable. This may be avoided by installing a 1M Ohm 1/4 watt resistor between pins 1 and 5 of the port B connector at one end of the cable (this needs only to be done on one of the terminals within the network). This will allow a charge on the shield to bleed into power ground. Table 8-1 lists the transmission lines.

Table 8-1 RS-485 Transmission Lines

PIN	SIGNAL
2	Data A
3	Data B
7	GND

NOTE
Make certain that Pin 7 (Isolated Ground) is always connected at both ends of the cable.

8.6 MULTIDROP COMMANDS

There are six commands specific to multidrop.

8.6.1 Select for Input

Function: Selects the terminal to receive all subsequent data and commands sent by the host computer.

Hazeltine sequence: 7EH 16H<addr>

ANSI sequence: ESC [7;<addr>p

where: ESC = 1BH
<addr>= terminal address (0-31)
(in Hazeltine mode, only the least significant 6 <addr> bits are tested)

After a terminal is selected, the host computer can transmit to it any characters or remote commands it would send normally. A terminal remains selected until another Select for Input command selects a different terminal. All characters/commands will be displayed or executed only by the selected terminal. An address of 00 selects all terminals.

8.6.2 Select for Output

Function: The selected terminal will return one record (containing all the data in its keyboard buffer) to the host computer.

Hazeltine sequence: 7EH 14H<addr>

ANSI sequence: ESC [6;<addr>p

where: ESC = 1BH
<addr>= terminal address (0-31)
(in Hazeltine mode, only the least significant 6 <addr> bits are tested)

A record consists of 0-126 characters followed by a Carriage Return and an EOT (04H). If the terminal receives the "Select for Output" command before the user types Enter or Return, the keyboard buffer record is not sent to the host. Only EOT (04H) is sent.

In Keyboard Character Mode, no "CR" is in the buffer unless it is the key typed by the user.

8.6.3 Enable Keyboard Echo

Function: This command allows the user to enable keyboard echoing.

Hazeltine sequence: 7EH 32H

ANSI sequence: ESC [=6h

where:

"h" is lower case H

The Enable Keyboard Echo is the default state.

8.6.4 Disable Keyboard Echo

Function: This command allows the user to disable keyboard echoing.

Hazeltine sequence: 7EH 33H

ANSI sequence: ESC [=6l

where:

"l" is lower case L

When this command is issued in multidrop mode, keys typed on the keyboard or keypad are not echoed on the screen. The key codes are put into the keyboard buffer and can be removed with the backspace key, but nothing is echoed to the screen.

The terminal's response to **Select for Output** is not affected by this command.

8.6.5 Enable Keyboard Character Mode

Function:

Hazeltine sequence: 7EH 34H

ANSI sequence: ESC [=7h

where:

 "h" is lower case H

Enable character mode works only when the terminal is in multidrop mode. When keyboard character mode is enabled in multidrop mode, a single key typed by the user completes a record.

A Carriage Return is not required. Each time the user types a key the terminal will send the keycode (followed by EOT) in response to a "Select for Output" command from the host. In this mode backspace will send the code for a backspace. Multiple keycodes will be sent in the record for those keys that send more than one keycode when typed.

8.6.6 Disable Keyboard Character Mode

Function:

Hazeltine sequence: 7EH 35H

ANSI sequence: ESC [=7l

where:

 "l" is a lower case L

Key codes are placed in the keyboard buffer until the record is completed by typing Enter or Return. The maximum length for a record is 127 bytes. Any characters received after 127 bytes are lost. A carriage return will still be needed to complete the record; however, the carriage return will not appear in the record when it is sent to the host. Keyboard character mode disabled is the default state.

8.7 STORED SCREEN UTILITIES MENU

NOTE

To enter or edit screens on the terminal, a keyboard is required. However, even without the keyboard, screens can be transmitted to a terminal and executed from a host computer.

The stored screen utilities menu is invoked from the main menu. The following menu options are provided:

- Edit** Enter a new screen or edit an existing screen. You will be prompted for the number of the screen to edit (1-255).
- Execute** Prompts the user for the number of the screen to be executed.
- Copy** One stored screen can be copied to another. You will be prompted for the source screen (to be copied) and the destination screen (into which the source screen will be copied). All data currently in the destination screen will be lost, replaced entirely by the data in the source screen. Copying from an empty screen is also a convenient way of emptying out the destination screen.
- Transmit** To another terminal or system. The user is prompted for the number of the screen to transmit (or all screens), and the number of the destination screen.

NOTE

If you choose to transmit all the screens, all 255 screens, including empty screens, will be transmitted. In particular, if empty screens are transmitted to another terminal they will erase screens of the same number already stored on the receiving terminal (see Appendix A).

- Directory** Displays the first line of each stored screen.
- Verify** Compares a downloaded image with the contents of a specified stored screen or all stored screens (refer to Appendix A). This option is useful if option 4 (Transmit Screen) was used to backup screen programs to a tape or to another computer system. Refer to Appendix A for instructions about storing screen programs on tape.
- Receive** Allows the terminal to receive screens from another terminal or system.

NOTE

Screens are transmitted, received, and verified over the printer port when in the "Stored Screen Utilities".

Two versions of each screen are included: the Hazeltine 1500 and the ANSI versions. (Each remote command exists in two forms, depending upon whether Hazeltine 1500 or ANSI emulation is desired.) To execute a stored screen written in Hazeltine 1500 format, the terminal must be configured for Hazeltine 1500 emulation. Likewise, to execute a screen written in ANSI format, the terminal must be configured for ANSI emulation.

Monochrome Screen Program Example

Hazeltine 1500

```
7E 09 01 12 05 36 0A ;Draw Box command<CR>
7E 11 13 06           ;Cursor to X,Y command to put cursor in box<CR>
7E 03 09 02           ;Change Char. Attributes command to quad,<CR>
                       ;blink, reverse<CR>
"WARNING"             ;message in box<CR>
```

ANSI

```
"<ESC>[2;1;6;19;11;55p" ;Draw Box command<CR>
"<ESC>[7;20H"           ;Cursor to X,Y command to put cursor in box<CR>
"<ESC>[1;9;2p"         ;Change Char. Attributes command<CR>
"WARNING"               ;message in box<CR>
```

Color Screen Program Example

Hazeltine 1500

```
7E 03 40 10           ;Select red on blue characters<CR>
7E 09 01 12 05 36 0B ;Draw a box<CR>
7E 11 13 06           ;Position cursor inside of box<CR>
7E 03 48 12           ;Select quad blinking, red on blue<CR>
"WARNING"<CR>
```

ANSI

```
"<ESC>[1;64;16p"       ;Select red on blue characters<CR>
"<ESC>[2;1;6;19;12;55p" ;Draw a box<CR>
"<ESC>[7;20H"         ;Position cursor inside of box<CR>
"<ESC>[1;72;18p"     ;Select quad blinking, red on blue<CR>
"WARNING"<CR>
```

If displayable characters or remote commands are encoded in ASCII, the sequence of ASCII characters must be enclosed in quotation marks ("). This is because all characters are interpreted as hex, unless enclosed in quotation marks. It is recommended that each separate line be enclosed in quotes, as in the above example. If the above example were enclosed in only two quotes, one at the beginning and one at the end, the <CR>s at the end of each line would be interpreted as remote commands and executed and the comments would be printed, thereby affecting the current cursor position. If each line is embedded in quotes, the <CR> at the end of each line is not interpreted as a remote command.

Note also that it is not required that Hazeltine-format screens be entered in hex and ANSI-format screens in ASCII. Hazeltine-format commands may also be entered in ASCII (if the commands are enclosed in quote marks), and ANSI-format commands in hex. However, since ANSI emulation is designed so that ASCII characters can be used, ANSI-format commands will typically be entered in ASCII.

Likewise, Hazeltine-format commands will typically be entered in hex. Note that commands entered as ASCII characters generally take up less program storage than hex codes, since it takes two hex digits to reproduce a single ASCII character.

The <CR> and <ESC> shown above actually take up only one character location apiece on the terminal screen. <CR> is visible so the user can see the end of the line.

Characters representing hex bytes (i.e., 7E 01 07) must always appear in pairs. For example, 1 is not a legal representation of the hex byte 01. The leading zero is required.

Spaces not enclosed in quotes are ignored. For example, either 7E 01 3E or 7E013E is an acceptable way of representing 3 hex bytes.

Any text appearing between quotes (") is stored as ASCII characters, and is used literally. All other text represents hexadecimal bytes. A double quote (") that appears between quotes will cause a single quote to be displayed when the screen is executed.

Any text following a semicolon (;) outside of quotes is a comment. Comments extend to the end of the line. When a stored screen is executed, comments are ignored.

It is a good idea to include an identifying comment as the first line of each stored screen so that it will show up in the directory entry for the screen.

8.9 NESTED SCREENS

The remote command `Execute Stored Screen` can be included within a screen program. For example, suppose screen `mm` contains an `Execute Stored Screen nn` command. When Screen `mm` is executed, all the remote commands will be executed in the order in which they occur. When the `Execute Stored Screen nn` command is executed, control will be transferred to Screen `nn`, which will be executed in its entirety. When Screen `nn` is finished, control will be returned to Screen `mm`, and execution of subsequent commands in screen `mm` will continue.

Any number of `Execute Stored Screen` commands can be included in one screen, and they may be included anywhere within the screen. Up to ten levels of nesting is permitted.

Stored Screens 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 screen, while each of the possible variations may be placed in separate screens. These variations can be called from other screens as needed. Then instead of redesigning each screen from scratch, the user may design a screen in parts and reuse these parts in other screens.

Two simple examples are provided below (one for Monochrome terminals and one for Color terminals). Suppose an application is monitoring a vat for two danger 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 letters "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 screens (see Figure 8-7).

- Screen 24: Creates the display "WARNING"
- Screen 20: Creates the display "TEMPERATURE TOO HIGH"
- Screen 21: Creates the display "PRESSURE TOO LOW"

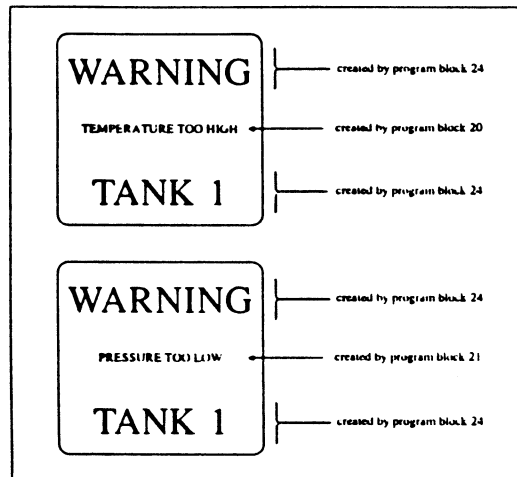


Figure 8-7 Nested Screen Example

To display the "TEMPERATURE TOO HIGH" warning, screen 7 would be executed.

To display the "PRESSURE TOO LOW" warning, screen 8 would be executed. Screen 7 contains an Execute Stored Screen command to execute Screens 24 and 20, while screen 8 executes Screens 24 and 21.

Monochrome Example

Screen 7 (For "TEMPERATURE TOO HIGH" Warning on Tank 1)

```
7E 10 18      ;Execute Stored Screen 18H (=24)
7E 10 14      ;Execute Stored Screen 14H (=20)
```

Screen 8 (For "PRESSURE TOO LOW" Warning on Tank 1)

```
7E 10 18      ;Execute Stored Screen 18H (=24)
7E 10 15      ;Execute Stored Screen 15H (=21)
```

Screen 24

```
7E 11 1D 03      ;Cursor to X,Y command
7E 03 1A 01      ;Blinking, double-size, high-intensity
"WARNING"
7E 11 1E 11      ;Cursor to X,Y command
7E 03 02 01      ;Double-size, high-intensity
"TANK 1"
7E 11 1B 0B      ;Cursor to X,Y command
7E 03 00 00      ;Default Character Attributes
```

Screen 20

```
"TEMPERATURE TOO HIGH"
```

Screen 21

```
"PRESSURE TOO LOW"
```

Color Example

Screen 7 (For "TEMPERATURE TOO HIGH" Warning on Tank 1)

```
7E 10 18 ;Execute Stored Screen 18H (=24)
7E 10 14 ;Execute Stored Screen 14H (=20)
```

Screen 8 (For "PRESSURE TOO LOW" Warning on Tank 1)

```
7E 10 18 ;Execute Stored Screen 18H (=24)
7E 10 15 ;Execute Stored Screen 15H (=21)
```

Screen 24

```
7E 11 1D 03 ;Cursor to X,Y command
7E 03 7A 11 ;Double-size, blinking, white on blue
"WARNING"
7E 11 1E 11 ;Cursor to X,Y command
7E 03 72 11 ;Double-size, blinking, white on blue
"TANK 1"
7E 11 1B 0B ;Cursor to X,Y command
7E 03 70 10 ;Regular size, white on blue
```

Screen 20

"TEMPERATURE TOO HIGH"

Screen 21

"PRESSURE TOO LOW"

Using the Serial Line for Input During Screen Program Execution

During execution of a screen program, the terminal monitors the RS-485 serial line, storing any characters received in a serial data buffer. When the last character in the screen program has been read, the terminal proceeds to read and execute the serial data buffer. The sample screens below use this feature to allow the host to select the next screen to be executed:

Monochrome Example

Screen 1

7E 1C	;Clear Screen
7E 11 1D 03	;Cursor to X,Y
7E 03 1A 01	;Change Character Attributes
"WARNING"	
7E 11 1E 11	;Cursor to X,Y
7E 03 12 01	;Change Character Attributes
"TANK 1"	
7E 11 1B 09	;Cursor to X,Y
7E 03 00 00	;Change Character Attributes
7E 10	;First part of Execute Screen command, ;waiting for last byte from host system

Screen 2

"TEMPERATURE TOO HIGH"

Screen 3

"PRESSURE TOO LOW"

Whatever character is received by the terminal over the serial line will determine the next screen to be executed. For example, if 02H is received, Screen 2 will be executed.

Color Example

Screen 1

```
7E 1C ;Clear Screen
7E 11 10 03 ;Cursor to X,Y
7E 03 7A 11 ;Double-size, blinking, white on blue
"WARNING"
7E 11 1E 11 ;Cursor to X,Y
7E 03 72 11 ;Double-size, white on blue
"TANK 1"
7E 11 1B 09 ;Cursor to X,Y
7E 03 70 10 ;Regular size, white on blue
7E 10 ;First part of Execute Screen command,
;waiting for last byte from host system
```

Screen 2

```
"TEMPERATURE TOO HIGH"
```

Screen 3

```
"PRESSURE TOO LOW"
```

Entering Variable Data into a Stored Screen

Nested screens can be used to display changing data in a screen display. For example, suppose that the message to be displayed on the screen is "Temperature of Tank 2 is xxx Degrees", where xxx varies. The following screens could display the message:

Screen 10

```
7E 1C ;Clear Screen
"Temperature of Tank 2 is "
7E 10 0B ;Execute Screen 11
" Degrees"
```

Screen 11

```
Contains a number written by the host
```

The host would periodically write new data to screen 11 by issuing a Receive Screen remote command. For example, to transmit the number "120" to screen 11, the host would transmit the following hexadecimal data to the terminal:

```
7E 1E 0B 22 31 32 30 22 7F
```

Note that 22 is the ASCII value of the quotation mark (").

Chapter 9

4800-E3 PARALLEL EXPANSION MODULE

9.1 -- INTRODUCTION

The 4800-E3 Expansion Module adds a parallel (digital I/O) interface port to the terminal.

9.2 CONFIGURATION MENU

The Parallel Port Configuration Menu is the second configuration menu in the 4800-E3. (The Main and Miscellaneous Configuration Menus are described in Chapter 3, as they are the same in all 4800-series terminals.) It is accessed by selecting the Configuration option from the Main Menu. It looks like the following:

```

-- Parallel Port Configuration Menu --

1      1=Parallel BCD           0=Parallel Binary
1      1=PC Sinks              0=PC Sources
1      1=High True Logic      0=Low True Logic
1      1=Strobed I/O          0=Timed I/O
02     Output Step Time in Milliseconds
02     Output Strobe Width in Milliseconds (Timed I/O)

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

Figure 9-1 Configuration Menu

9.3 COMMUNICATIONS STATUS REGISTER

There is no dedicated Communication Status Register in the 4800-E3.

9.4 CABLING THE PARALLEL PORT

After installation, the terminal is provided with a parallel (digital) input/output port for communicating with a programmable controller.

The back of the terminal has a label which has a map of the pinouts for the parallel input and output ports. This label is reproduced in Figure 9-2 for reference. Arrows pointing inward indicate inputs, while those pointing outward indicate outputs.

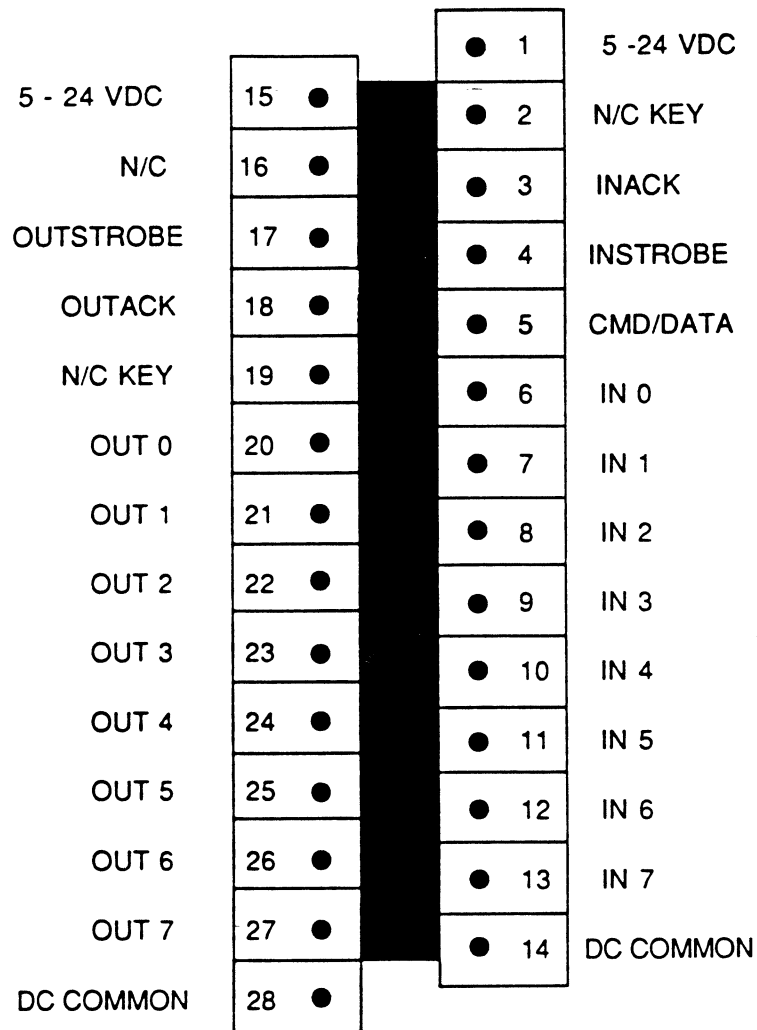


Figure 9-2 Parallel I/O Port Pinouts

The 4800-E3 package contains two connectors (along with two keying pins) which mate with the two connectors on the parallel port. These connectors and keying pins have the following part numbers:

- Phoenix MSTB 1.5/14-ST-5.08 17 62 06 2 (connector)
- Phoenix CS-MSTB (keying pin)

These are readily available from outside vendors.

The parallel port can be connected to programmable controller digital input and output signals whose "high" voltage is in the range of 5 to 24 volts DC. To maintain electrical isolation between the programmable controller and the terminal, the DC excitation voltage for the interface is provided by an external, user-supplied power source (the programmable controller's DC supply is typically used).

The Input Data and Output Data lines can be used as "high true" or "low true" signals. The choice of which is to be used is selected in one of the terminal's set-up menus (see Chapter 3). The high/low true selection affects only the IN and OUT lines. You also have a choice whether the terminal sources or sinks current on these interface signals. The selection of the source/sink affects only the input signals of the parallel port. (Figure 9-2 indicates which signals are input and which are output.)

The voltage at which a signal is sensed as a logic "TRUE" or "FALSE" is dependent on several factors. Table 9-1 explains these relationships.

Table 9-1 Parallel Port Electrical Specifications

INPUTS

TTL-level Inputs are TTL compatible when the input power supply is set for 5V + 5% and the inputs are in the sinking mode (i.e., if connected to TTL programmable controller I/O module). Adhere to TTL standards. These include:

Voltage under 0.8V = 0

Voltage over 2.0V = 1

I_I = 1.6mA at 0.4V

I_{IH} = Current is not required into the input for a logic high.

Non-TTL Minimum hysteresis is 2% of the supply voltage.

Sinking Mode

$I_{IL} = \frac{V_1 - V_{IL}}{2700}$, where V_1 is the input port power supply (pin 1).

I_{IH} = Input current is not required for a logic high.

V_{IH} = 40% or more of power supply voltage.

V_{IL} = 25% or less of power supply voltage.

Sourcing Mode

$I_{IH} = \frac{V_{IH}}{2700}$

I_{IL} = No current is required to be sunk from the input for a logic low.

V_{IH} = 75% or more of power supply voltage.

V_{IL} = 60% or less of power supply voltage.

Hysteresis

2% of supply voltage

OUTPUTS

$V_{OH} = V_{CC} - .75V$ min. @ -10mA max.

$V_{OL} = .4V$ max. @ 30mA max.

Figure 9-3 shows typical wiring between a terminal equipped with a 4800-E3 and a programmable controller.

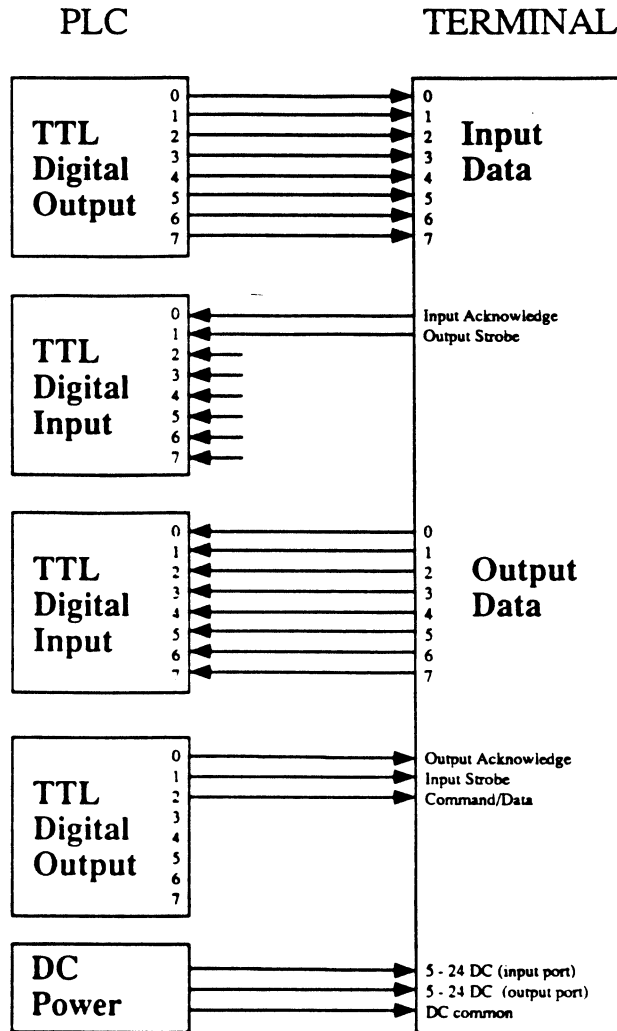


Figure 9-3 Typical Wiring: Terminal to PLC

Strobe lines should be individually shielded, twisted pairs, where ground is one of the wires of the pair. Data inputs and outputs should be in separate cables, so that data inputs have a single ground return, and shield and data outputs have a single ground return and shield. All shields should be connected to the PLC chassis ground only.

Maximum cable length is 50 feet.

9.5 SPECIAL COMMANDS

There are no special PLC-associated commands in the 4800-E3.

However, there are two types of remote commands: parallel and serial. Parallel and serial remote commands are similar in function. The biggest differences between them are parallel commands are received on the parallel port and serial commands on the serial port.

The serial remote commands are described in Chapter 6.

9.6 DATA TRANSFER BETWEEN A PROGRAMMABLE CONTROLLER AND THE PARALLEL PORT

Wiring between the terminal and a programmable controller is described in Section 9.x.

9.6.1 Parallel Input Port

Commands and data are passed from the host to the terminal through the parallel input port. This section describes the timing constraints for the parallel input port.

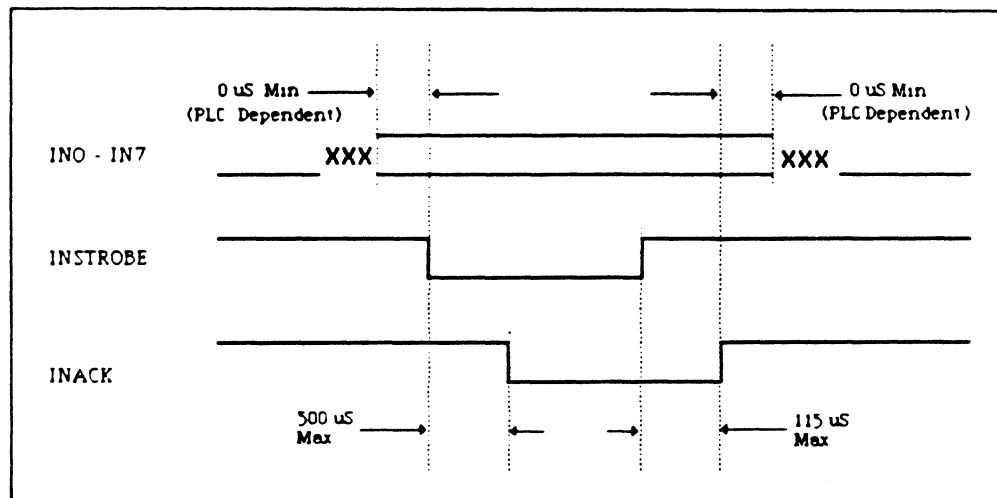
The input port consists of 10 input lines and one output line. The lines are used as follows:

- 8 data (inputs IN0-IN7)
- 1 command/data select (input CMD/DATA)
- 1 input strobe (input INSTROBE)
- 1 ready - terminal ready for command or data (output INACK)

Input may be done either with strobed or timed data transfer.

Strobed Transfer

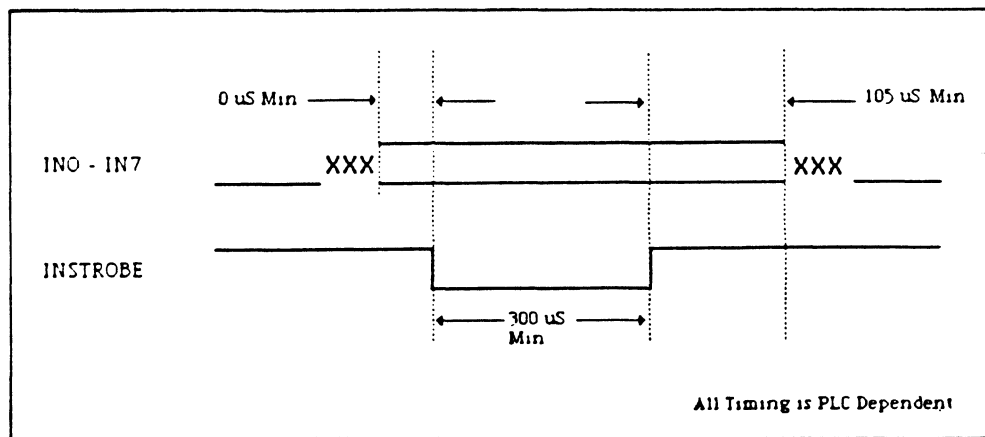
The strobed transfer is done using the INACK and INSTROBE lines as follows:



1. The terminal hardware indicates it is ready for a command or data byte by raising the INACK line.
2. The host programmable controller sets up the data lines and the CMD/DATA line (0 for data, 1 for command) and then lowers the INSTROBE line. Note that whether 0 is high or low depends upon whether the PC was configured for source or sink in the Miscellaneous Configuration Menu. If the terminal is configured for source, 0 is interpreted as high voltage. If the terminal is configured as sink, 0 is interpreted as low voltage.
3. The terminal lowers the INACK line after the leading edge of the INSTROBE pulse. After the programmable controller sees INACK go low, it should raise INSTROBE.
4. The terminal responds to the raising of INSTROBE by reading the parallel data port. This raises the INACK line, allowing the programmable controller to begin another transfer.

Timed Transfer

The timed transfer is done using the INSTROBE line as follows:



1. The host programmable controller sets up the data lines and the CMD/DATA line and then lowers the INSTROBE line (minimum of 30 μs).
2. The host programmable controller must hold the INSTROBE line high before beginning the next data transfer.
3. At this point, the terminal has accepted the data at the parallel data port.

9.6.2 Parallel Output Port

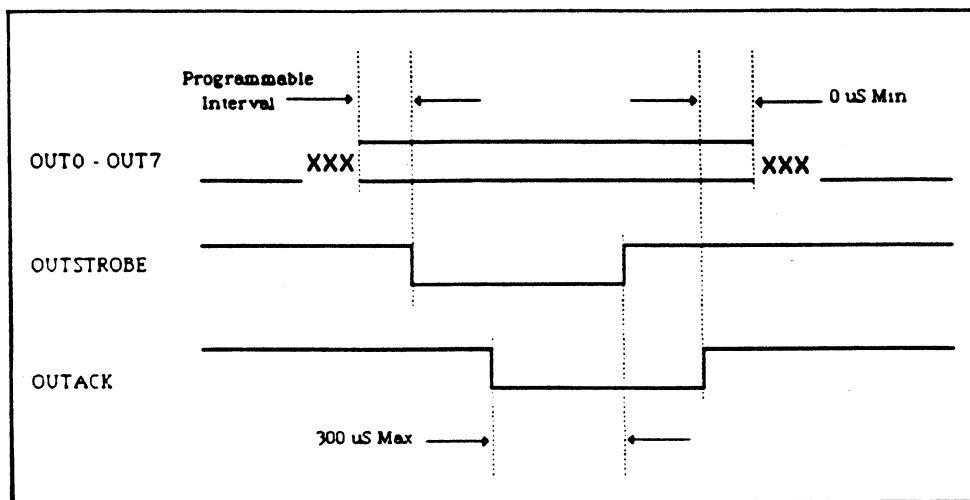
The parallel output port consists of 9 output lines and one input line. The lines are used as follows:

- 8 data (outputs OUT0-OUT7)
- 1 ready - Host ready for data (input OUTACK)
- 1 data available (output OUTSTROBE)

Output may be done either with strobed or timed data transfer.

Strobed Transfer

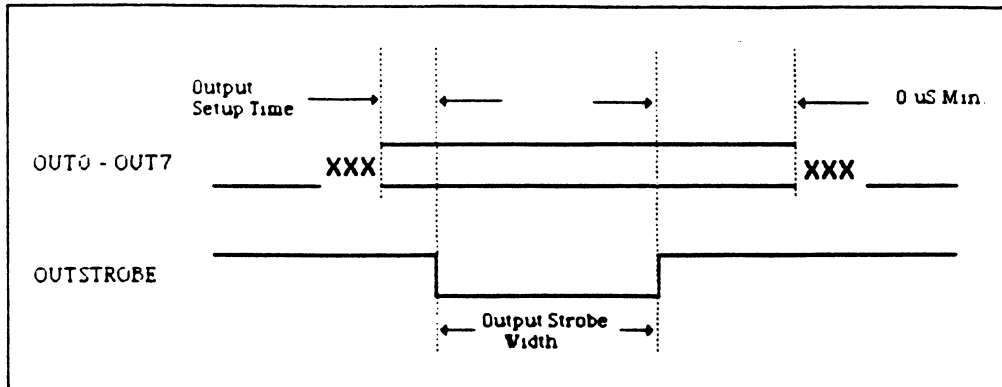
Strobed transfer is done using the OUTACK and OUTSTROBE lines as follows:



1. The host programmable controller indicates it is ready for a data byte by raising the OUTACK line.
2. The terminal sets up the data lines and then lowers the OUTSTROBE line when the data is fully available (after Data set-up time has expired.)
3. The host programmable controller lowers the OUTACK line after the leading edge of the OUTSTROBE pulse. After the terminal sees OUTACK go low, it raises OUTSTROBE.
4. The host programmable controller sees OUTSTROBE go high and raises OUTACK in response. This informs the terminal that another data transfer can begin.

Timed Transfer

Timed transfer is done using the OUTSTROBE line as follows:



1. The terminal sets up the data lines, waits for a time interval (determined by the Output Setup Time in the Configuration Menu #2), and then drops the OUTSTROBE line.
2. The terminal holds OUTSTROBE low for a time interval (determined by the Output Strobe Width in the Configuration Menu #2).
3. The terminal raises OUTSTROBE and holds it high before beginning the next data transfer.

A 256 byte FIFO is used to buffer data output from the terminal.

9.7 PARALLEL REMOTE COMMANDS

Parallel commands sent to the terminal over its parallel input port have a numeric format. Table 9-1 lists this format.

Table 9-2 Parallel Remote Commands

COMMAND	FORMAT
Reset	00
Execute Program Block	<block#>01
Transfer From Data Register	<reg#>02
Transfer To Data Register	<reg#><low byte><high byte>03
Short Transfer From Data Register	<reg#>04
Short Transfer To Data Register	<reg#><low byte>05
Execute Subprogram Block	<block#>06

The numbers 00 through 06 are all single bytes.

<block#> = any (single-byte) number from 01 to FF (binary) or 01 to 99 (BCD)

<reg#> = any (single-byte) number from 00 to FF (binary) or 00 to 99 (BCD)

<short reg#> = any (single-byte) number from 00 to FF (binary) or 00 to 99 (BCD)

<low byte> = any (single-byte) number from 00 to FF (binary) or 00 to 99 (BCD)

<high byte> = any (single-byte) number from 00 to FF (binary) or 00 to 99 (BCD)

When transmitting a command to the terminal, first the data byte(s) must be strobed in, then the command byte.

Whether the terminal interprets the parameters of the parallel command as BCD or binary depends upon the "Parallel BCD/Binary" item is set in the Configuration Menu.

A queue is used to buffer all data bytes received over the parallel port. Command bytes are put in a separate queue. The command buffer will still accept commands when it is full. This allows a terminal to execute a Reset command whenever it is received. (However, if the command queue is full when another command is received, the last command received will be ignored.)

9.7.1 Reset

Description: The Reset command causes the terminal to initialize itself. If the data registers are selected to be in volatile RAM, they are set to zero. All data input and output queues are cleared. Finally, the start-up program block (if non-zero) is executed.

Syntax:
00

The Reset command will be performed whenever it is received over the parallel 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.

9.7.2 Execute Program

Description: This command causes the terminal to terminate any program currently executing, and to begin execution of the specified program.

Syntax:
<block#>01

9.7.3 Transfer From Data Register

Description: This command causes the terminal to return two bytes of data via the parallel port indicating the current value of the specified register. The data is transferred <low byte><high byte>.

Syntax:
<reg#>02

9.7.4 Transfer To Data Register

Description: This command causes the specified value to be placed into the specified data register.

Syntax:
<reg#><low byte><high byte>03

9.7.5 Short Transfer From Data Register

Description: This command sends only the lower byte (LSB) of the specified register out the parallel port.

Syntax:
<reg#>04

9.7.6 Short Transfer To Data Register

Description: This command causes a single byte to be placed into the lower byte (LSB) of the specified data register. The high byte (MSB) is set to zero.

Syntax:
<reg#><low byte>05

Chapter 10

4800-E4 SERIAL EXPANSION MODULE

10.1 INTRODUCTION

After installation of the 4800-E4 Expansion Module, there is an additional serial port resident in the terminal. The port located on the expansion board is defined as the "primary" port for communications with the control system. The original port on the terminal controller board is considered as a "secondary" port for general purpose I/O. Thus, the "primary" port is used to receive all of the control system commands sent to the terminal, and the "secondary" port is generally used for two purposes:

- Host to serial port output -- the control system orders the terminal to write data out the secondary port. Used for printer, data logging, etc. Could be used to control any serial device from the host control system.
- Backup and restoration of screen programs and data registers. The terminal can read programs from the serial port and write programs out the serial port.

The signal definitions are provided in Section 10.4.

10.2 CONFIGURATION MENU

The Serial and Miscellaneous Configuration Menus are described in Chapter 3, as they are the same in all 4800-series terminals. The Primary Port Configuration Menu is accessed by selecting the Configuration option from the Main Menu. It is identical to the Serial Port configuration Menu except in the title of the menu at the top of the screen. See Chapter 3 for details.

10.3 COMMUNICATIONS STATUS REGISTER

There is no dedicated Communication Status Register in the 4800-E4.

10.4 CABLING THE SERIAL PORTS

Both the primary and secondary serial ports use standard RS-232 signals. These are given in Table 10-1 for reference. For both serial ports the terminal provides 300 volt isolation between pins 2, 3, and 7 (Transmit Data, Receive Data, and XMIT/RCV Ground), and its internal logic on the secondary serial port only. To maintain this isolation, the modem control lines should not be connected. If, however, the modem control lines are needed, pins 9 and 10 on the RS-232C port connector must be connected together. In this case the isolation is lost. According to RS-232C specifications, the cable between two RS-232C serial devices is limited to 50 feet in length.

Table 10-1 RS-232C Signal Definitions

PIN NUMBER	SIGNAL	FUNCTION
1	GND	Frame Ground
2	TD	Transmit Data
3	RD	Receive Data
4	RTS	Request To Send *
5	CTS	Clear To Send *
6	DSR	Data Set Ready (DSR) *
7	GND	TD/RD Ground
9	GND	Modem Control Ground **
10	GND	TD/RD Ground **
20	DTR	Data Terminal Ready * ***

NOTES:

- * Modem Control
- ** 9 & 10 must be connected when using modem control lines
- *** Not available on Primary Port

10.5 SPECIAL COMMANDS

There are no special PLC-associated commands in the 4800-E3.

The serial remote commands are described in Chapter 6.

Chapter 11

4800-E5: MULTIDROP EXPANSION MODULE

11.1 INTRODUCTION

The 4800-E5 Expansion Module adds screen memory, TOD clock, two RS-485 serial ports, and OIL to the terminal.

The RS-485 ports are designed to be used to network a number of Xycom terminals in a multidrop network. This network is described later.

11.2 CONFIGURATION MENU

The configuration menus are called up from the Main Menu (discussed in Chapter 3). The Multidrop Configuration Menu looks like the following:

```

-- Multidrop Port Configuration Menu --

6 Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600 7=19.2K
0 Parity - 0=even 1=odd
0 1=Parity Enabled          0=Disabled
0 1=8 Data Bits            0=7 Data Bits
Multidrop Terminal Address (1-31)
Record Terminal Byte (0-255)
Response Delay in Milliseconds

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

Figure 11-1 Configuration Menu

Baud Rate. The baud rate of the serial channel should be set to match that of the serial device to which the terminal is connected.

Parity. There are two parity options. The first will set the parity used in communication on the Multidrop network to either odd or even. The type selected should match the other communication device(s). The second will enable or disable the parity.

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

Multidrop Terminal Address. This sets the address of this terminal in the range 1 to 31 (decimal) for input or output in multidrop mode. An address of 0 disables the terminal from the multidrop network.

Record Termination Byte. Each time the 4800-E5 equipped terminal responds to a Select for Output multidrop command, it will append this character to the end of the message. This character may be selected for the most convenient programming of the multidrop master. This character is specified as the decimal value of that ASCII code (e.g., 013 = ASCII Carriage Return).

Response Delay Time. The "Response Delay Time" option is used only when the terminal is in multidrop mode. This option specifies the minimum amount of time the terminal will wait before transmitting data to the host. This gives the multidrop master adequate time to disable its RS-485 transmitter and enable its receiver so it can receive the data from the terminal.

11.3 SERIAL LOOP BACK TEST

The ports on the terminal can be tested from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plug used on the serial port. The test set-up used for the RS-485 ports is shown later.

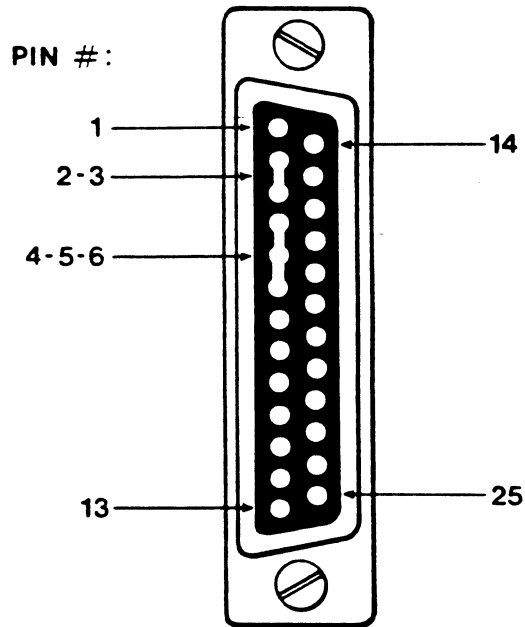


Figure 11-2 Serial Port Test Plug

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

Controller port time out error.

Controller port data error.

Controller port CTS-RTS failure.

Controller port DTR-DSR failure.

RS-485 Test

The test of the RS-485 multidrop serial port requires an independent 4800-MDA (MultiDrop Adapter) unit. A cable must be run from the 4800-E5 to the Xycom 4800-MDA and back. Figure 11-3 shows Test Cable 1, from terminal RS-485 port A to the 4800-MDA port A. Figure 11-4 shows Test Cable 2, back from the 4800-MDA port B port to the terminal RS-232 port.

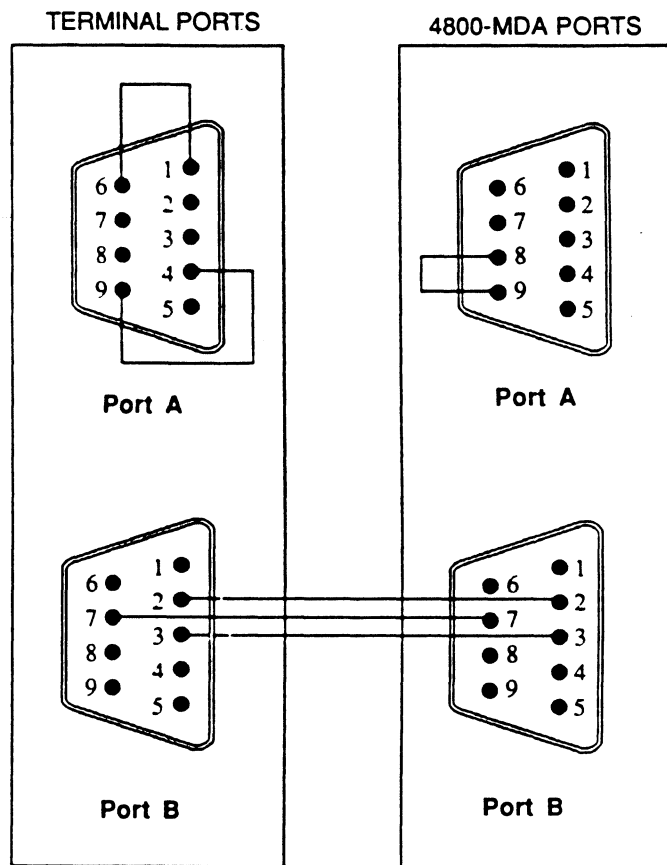


Figure 11-3 Test Cable 1

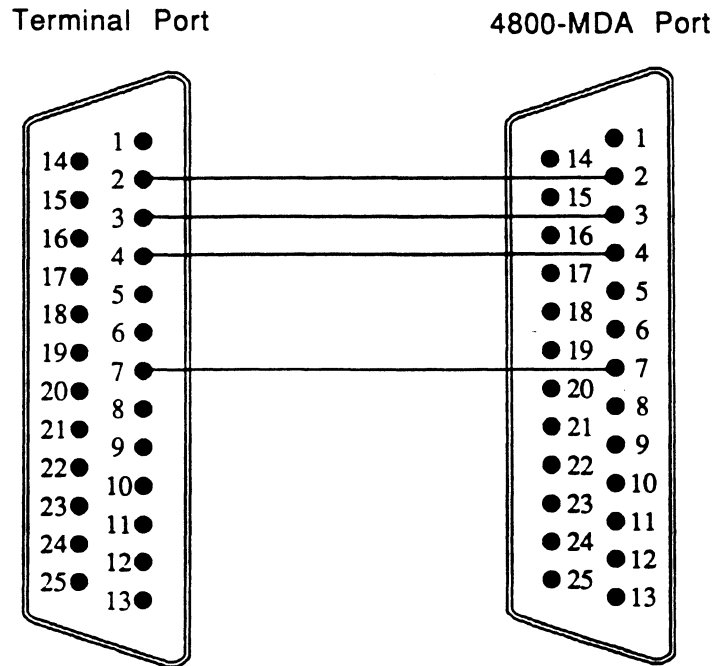


Figure 11-4 Test Cable 2

With the construction and installation of these two Test Cables, the RS-485 diagnostics test can be run by selecting #9 from the diagnostics menu.

If the Diagnostics test runs without any problems, the message "RS-485 Test Passed." will appear on the terminal. The following are valid error messages:

- 4800-E5 UART failed to receive any data.
- 4800-E5 UART received incorrect data.
- Communications Module timeout circuit failed.
- Communications Module timeout circuit failed to reset.
- Controller UART failed to receive any data.
- Controller UART received incorrect data.
- 4800-E5 timeout circuit failed.
- 4800-E5 timeout circuit failed to reset.

11.4 MULTIDROP NETWORK OVERVIEW

The Xycom RS-485, positioned at some point in a network, can communicate with up to 31 terminals. Messages and data are passed from the host computer simultaneously to all of the terminals in the network.

The positions, or terminals, along the network are called nodes. Each node in the network has a unique address (set in the terminal Configuration Menu). Each message or command transmitted by the host is prefaced by the address of the terminal for which it is intended. An addressed terminal can also transmit a message back to the host computer.

The network interconnection can be made from either port on the RS-485, as they are designed in parallel. For correct operation, each end of the network must be terminated. This means that specific resistor values must be connected between the wires of the cables and ground. These terminations are designed into Xycom products; only certain pins need be tied to accomplish the termination.

The Xycom multi-drop network uses a half-duplex "poll/select" protocol. Poll/select refers to the fact that the host computer controls all communications by issuing commands to the terminals. The host can transmit data to any or all of the terminals at one time.

A terminal will transmit data only in response to the host computer. For this to happen, the terminals are polled, one by one or specifically, with a data request from the host. If the terminal has data, it will transmit. If not, it will send a negative reply.

To send data to a terminal, the host will send a **Select for Input** command to a specific terminal. This terminal will accept all data and execute all commands until another command is sent from the host, either to it or another terminal.

To read data, the host computer will issue a **Select for Output** command. The host will periodically send this command to each terminal on the network to check for data.

The host computer must assert an RTS or CTS in order to begin transmission. If more than 37.6 msec elapses between start-bits, the RS-485 interface will return to the receiving state. If this happens, the user must de-assert the signal, then assert it again. This is a fail-safe feature designed into the RS-485 that will not allow more than one transmitter to be active at one time and/or prevent faults at network modes from disabling the system.

11.4.1 Multidrop Commands

To send commands or data to a terminal, the host computer first issues a **Select for Input** command to the terminal. This tells a terminal it should accept all commands and data until another terminal is selected. Then the host computer can send any characters or remote commands (for example, **Draw Box** or **Set Attributes**), which it would normally send to a terminal. All such commands or data will be received and executed by the selected terminal until a **Select for Input** command selects another terminal.

To read data from a specific terminal, the host computer issues a **Select for Output** command to the terminal. In response, the terminal will send one record back to the host computer. A record is a complete line typed on the terminal (0-126 characters followed by the ENTER or RETURN key). If a complete line has been typed on the specified terminal, the terminal will transmit all the characters followed by a Carriage Return and an EOT character. If a complete record (line) is not available, the terminal will transmit a single EOT character. The host computer software will periodically issue **Select for Output** commands to each terminal on the line to check if any has data to transmit.

11.4.2 Operator Input

Keys typed by the user are echoed on the screen and put into the keyboard buffer. The terminal can hold one complete record, i.e. it will wait until it is transmitted then accept any part of another record (the maximum length for a record is 127 bytes). The Enter or Return key terminates the current keyboard buffer record. Before the Enter or Return key is pressed, the operator can use the backspace or left arrow key to delete erroneous keystrokes both from the terminal internal data buffer and from the screen.

Once the Enter or Return is typed, no more keyboard input is accepted until the host reads the keyboard buffer record with the **Select for Output** command. If the terminal receives the **Select for Output** command before the user types Enter or Return, the keyboard buffer record is not sent to the host. Only the EOT (04H) is sent.

11.4.3 Keyboard Echo and Keyboard Character Mode

There are two commands in multidrop mode that allow the Keyboard Echo to be Enabled or Disabled. Keyboard echo enabled is the default state. When Keyboard echo is Disabled in multidrop mode (disable Keyboard echo works only when the terminal is in multidrop mode), keys typed by the user are not echoed on the screen. The keycodes are put into the transmit buffer and can be removed with the backspace key, but nothing is echoed on the screen. If the terminal receives the **Select for Output** command before the user types Enter or Return, the keyboard buffer record is not sent to the host. Only the EOT (04H) is sent.

The Enable Keyboard Character Mode works only when the terminal is in multidrop mode (Keyboard Character Mode Disabled is the default state). When Keyboard Character Mode is Enabled in multidrop mode, a single key typed by the user completes a record. No carriage return is required. Each time the user types a key the terminal will send the keycode (followed by EOT) in response to a **Select for Output** command from the host. In this mode backspace will send the code for a backspace. Multiple keycodes will be sent in the record for those keys that send more than one keycode when typed.

11.4.4 Terminal Addressing and Broadcasting

Each terminal has a unique address assigned by the user from the terminal keyboard through the terminal configuration menu. Valid terminal addresses are 1-31 (base 10).

A host computer can select all terminals for input (a.k.a. broadcasting) by issuing a **Select for Input** command accompanied by address 00. Until another **Select for Input** is issued, all terminals will accept commands from the host. This allows the host to broadcast commands and data to all of the terminals simultaneously.

When selected at the terminal, address 00 disables the multidrop protocol and allows the terminal to function as a non-multidrop terminal (i.e., no select for input or output), using the RS-485 port.

Only the lower 6 bits of the address sent in the **Select for Input** and **Select for Output** commands are used for the address comparison. The upper two bits are ignored. This gets around the problem of trying to send control codes as addresses in Hazeltine mode.

11.4.5 Multidrop System Considerations

When selected for output, a terminal could respond with an EOT (04H) or the first byte of the keyboard buffer record in as little as 60usec.

The response to the **Select for Output** could take longer if the terminal was recently selected for input and it was still executing the commands it received at that time.

In any case, a terminal that is on-line will always respond to the **Select for Output** command. If a terminal does not respond in a fixed amount of time, the host software should assume the terminal is off-line.

The time-out time should be carefully chosen. If the host does not wait long enough before timing-out it may assume an on-line terminal is off-line and start communicating with other terminals on the RS-485 network. When the terminal thought to be off-line finally responds to the **Select for Output** command, it may corrupt the current communications on the network.

The "Minimum Reply Delay Time" option in the Configuration Menu can be used to specify the minimum time the terminal will wait before transmitting data to the host after receiving a "Report" type command or a "Select for Output" command. This gives the host time to disable its RS-485 transmitter and enable its receiver so it can receive the data from the terminal.

Keys that return control codes will have their codes displayed on the screen using a two-character representation of their ASCII names. The control codes are stored "as is" in the keyboard buffer. If the terminal is in quad-size, double-high, or double-size mode, the control codes will be displayed on the screen as spaces.

11.4.6 Using Remote Commands With a Multidrop Network

As previously stated, any of the remote commands described in your terminal manual can be transmitted to selected terminals in the Multidrop Network. However, there are some remote commands which must be given consideration when programming the host computer.

If a terminal is selected for input, and it receives either the command to Return Cursor Position or the command to Transmit Stored Screen, the terminal will return the requested data to the host computer immediately (i.e. without waiting to be selected for output). This means if the host computer issues either of these commands, it must be ready to receive data from the selected terminal immediately. This situation would also occur if any command to return data to the host were to be executed from within a screen program while a terminal is on the Multidrop Network. Therefore, it is recommended those commands which are designed to return data to the host not be imbedded in any screen programs, unless the host computer is ready to receive data at any time while the screen program is executing.

Likewise, the two multidrop commands (Select for Input and Select for Output) should never be executed from screen programs while a terminal is in multidrop mode. These commands are intended to be issued from a host serial port within the network and they could cause a conflict in the communication protocol.

In addition, while a screen program is executing, a terminal can still receive data and commands. However, commands and data are buffered and are not examined by the terminal until the screen program finishes. Thus, if a terminal is executing a screen program when the host issues it a Select for Output command, the terminal will not respond until it finishes the screen program (by which time the host might have already "timed out"). This situation should be considered when programming the host computer.

11.5 CONNECTING A MULTIDROP NETWORK

Each RS-485 multidrop interface on a Xycom Multidrop Network has 2 ports: port A and port B. The internal configuration of the ports is such that both ports on a terminal are wired in parallel. Therefore, the ports on any given terminal may be used interchangeably in terms of installing the incoming and outgoing network lines. However, if a terminal is at the head-end or tail-end of a network, the jumper plugs used to facilitate termination can only be installed on port A. Figure 11-5 illustrates a typical hook-up for a 3-terminal multidrop network.

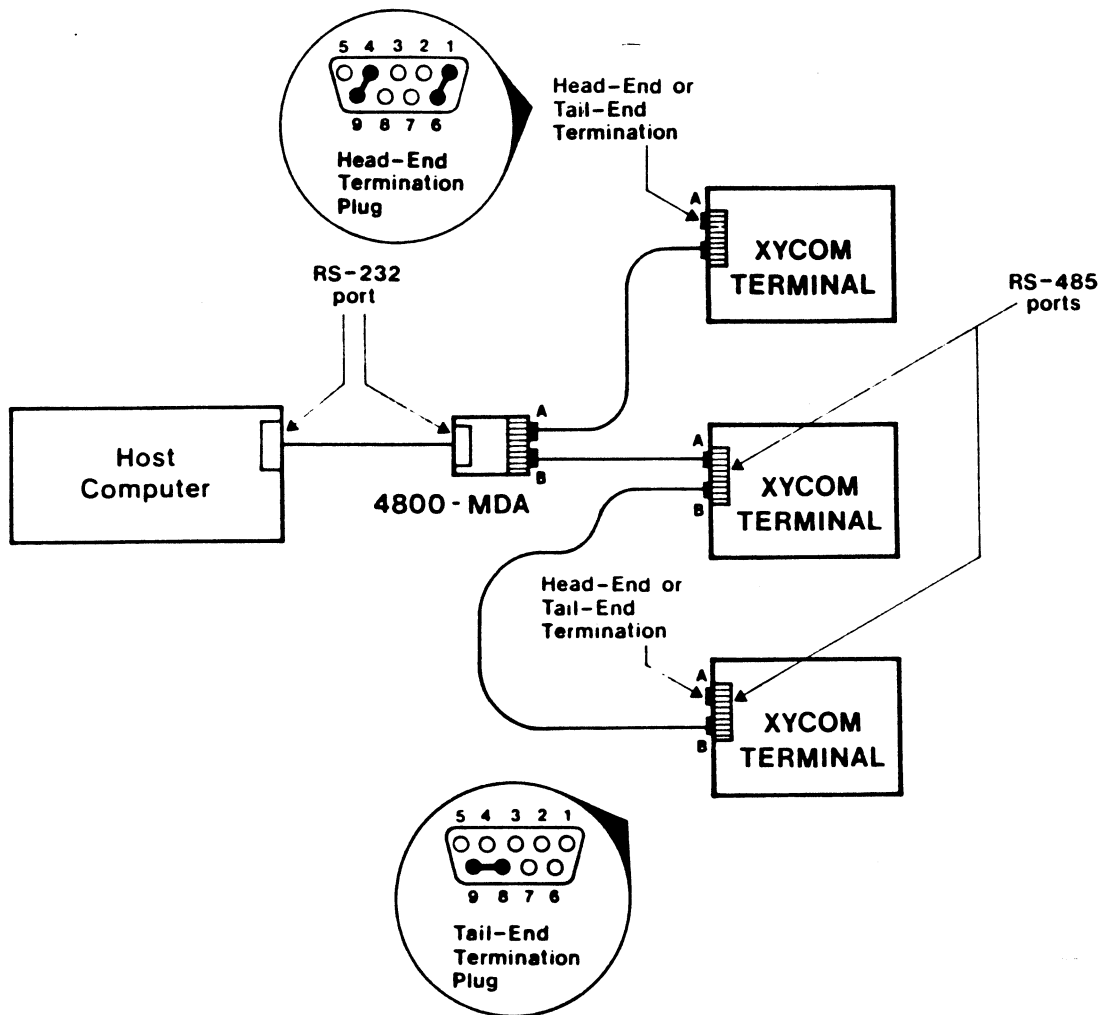


Figure 11-5 Multidrop Connection

Only pins 2, 3, and 7 are used on each port of the RS-485 interface. Note the termination plugs on port A of the 4800-E5 Expansion Module (head-end) and port A of the last terminal in the network (tail-end).

All Xycom RS-485 compatible networking devices are designed with internal termination so they may be used at either the head-end or tail-end of a network. To activate the termination circuitry one need only install jumpers at specific pins on port A of the head-end and tail-end nodes. This is best facilitated by installing a male DB-9 plug on port A of the corresponding device.

11.5.1 Head-end Termination

On port A of the head-end, install a male DB-9 plug with pins 1 & 6 and 4 & 9 shorted together (refer to Figure 11-5). Installing such a jumper plug will terminate the head-end of the network with the proper RS-485 impedance, as well as provide an offset on the line so that during periods of no transmission, all receivers see a marking condition.

11.5.2 Tail-end Termination

On port A of the tail-end, install a male DB-9 plug with pins 8 & 9 shorted together (refer to Figure 11-5). Installing such a plug will terminate the tail-end with the proper RS-485 impedance.

11.5.3 RS-485 Cabling

In situations where cable length is considerable or in electrically noisy environments, a static charge might build up on the shielding of the cable. This may be avoided by installing a 1M Ohm 1/4 watt resistor between pins 1 and 5 of the port B connector at one end of the cable (this needs only to be done on one of the terminals within the network). This will allow a charge on the shield to bleed into power ground. Table 2-1 lists the transmission lines.

Table 11-1 RS-485 Transmission Lines

PIN	SIGNAL
2	Data A
3	Data B
7	GND

NOTE
Make certain that Pin 7 (Isolated Ground) is always connected at both ends of the cable.

11.6 MULTIDROP COMMANDS

There are six commands specific to multidrop. (NOTE: Any of the other remote commands described in your terminal manual as well as the additional remote commands listed in Chapters 4 and 5 of this manual, can be transmitted to the selected terminal(s) in the Multidrop Network.)

11.6.1 Select for Input

Function: Selects the terminal to receive all subsequent data and commands sent by the host computer.

Hazeltine sequence: 7EH 16H<addr>

ANSI sequence: ESC [7;<addr>p

where: ESC = 1BH
<addr>= terminal address (0-31)
(in Hazeltine mode, only the least significant 6 <addr> bits are tested)

After a terminal is selected, the host computer can transmit to it any characters or remote commands it would send normally. A terminal remains selected until another Select for Input command selects a different terminal. All characters/commands will be displayed or executed only by the selected terminal. An address of 00 selects all terminals.

11.6.2 Select for Output

Function: The selected terminal will return one record (containing all the data in its keyboard buffer) to the host computer.

Hazeltine sequence: 7EH 14H<addr>

ANSI sequence: ESC [6;<addr>p

where: ESC = 1BH
<addr>= terminal address (0-31)
(in Hazeltine mode, only the least significant 6 <addr> bits are tested)

A record consists of 0-126 characters followed by a Carriage Return and an EOT (04H). If the terminal receives the "Select for Output" command before the user types Enter or Return, the keyboard buffer record is not sent to the host. Only EOT (04H) is sent.

In Keyboard Character Mode, no "CR" is in the buffer unless it is the key typed by the user.

Chapter 12

4800-E6: ALLEN-BRADLEY DATA HIGHWAY

12.1 INTRODUCTION

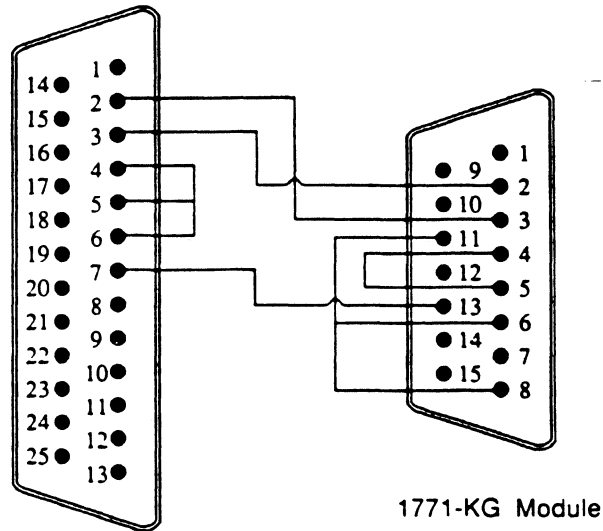
This chapter contains all of the information that is specific to the 4800-E6 Allen-Bradley (A-B) Data Highway PLC interface. The protocol to be used by the module to communicate with the Data Highway is a Full-duplex, asynchronous, point-to-point protocol that closely conforms to the ANSI X3.28 standard.

The specific commands that allow communications are listed in Section 12.6. There they are discussed in detail, including examples. These commands are:

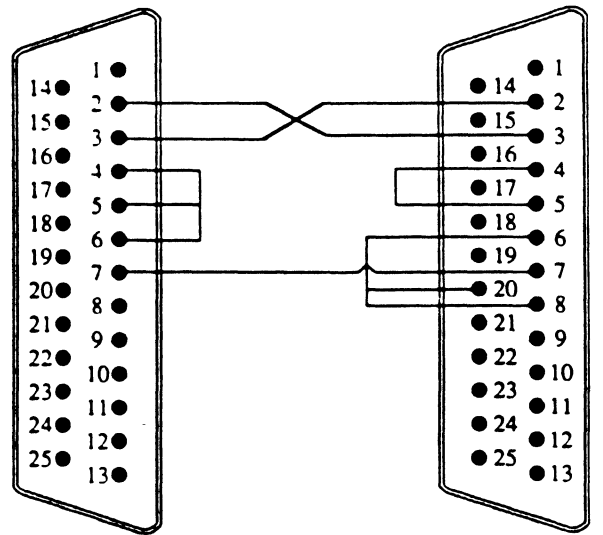
GET
GETDIAG
GETIO
PUT
PUTIO

12.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC and/or network through a cable. This is not a standard cable. The connectors must be fabricated. Figure 12-1 below illustrates the connections that must be made between the Expansion Card port and the PLC.



4800-E6



4800-E6 Module

KF2 Module

Figure 12-1 Connections

The 4800-E6 can be attached to one PLC or to a Data Highway Network. The possibilities are shown below.

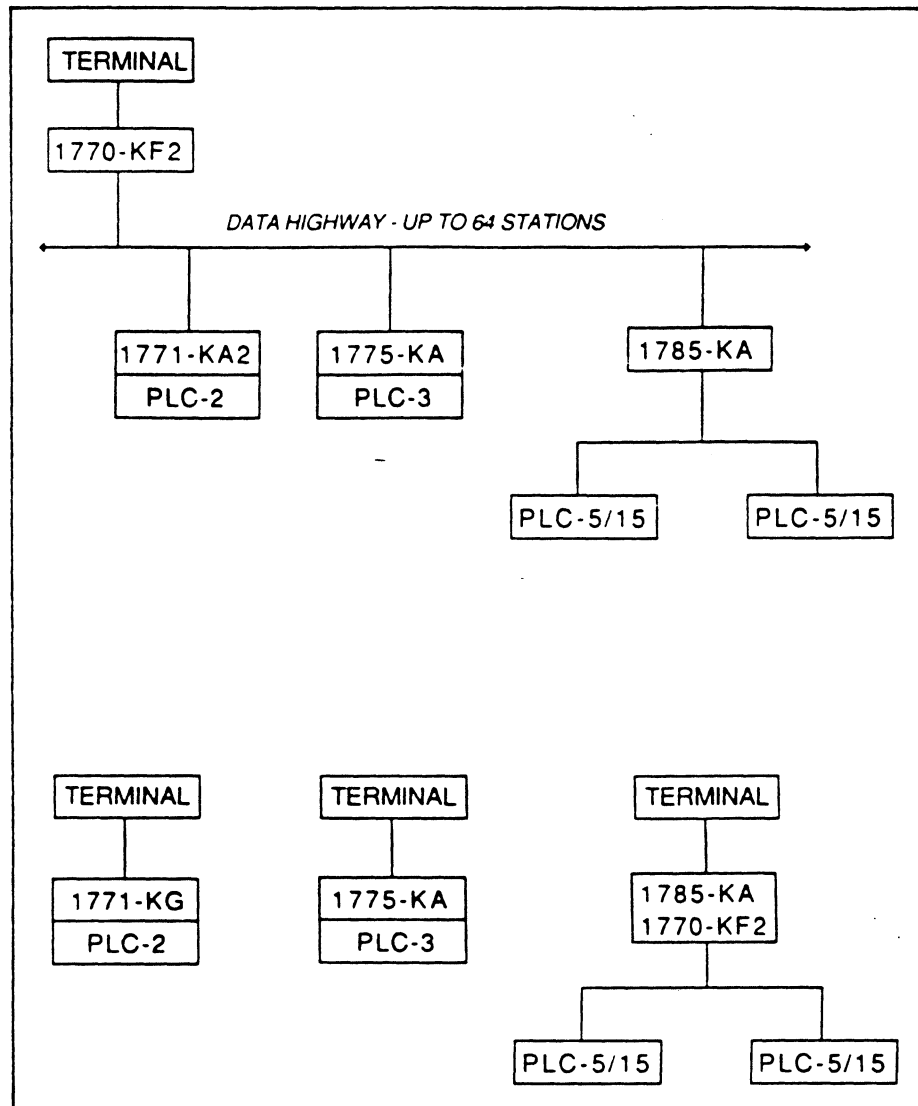


Figure 12-2 Configurations

12.3 CONFIGURATION MENU

The configuration menus are called up from the Main Menu (discussed in Chapter 3). The Data Highway Menu looks like the following:

```

-- Data Highway Port Configuration --

6 Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600
0 Parity - 0=none 1=even
02 Timeout value: 2 - 40 (tenths of a second)
0 Mode - 0=Foreground 1=Background
000 Station Number (NNN Octal)

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

Figure 12-3 Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Parity. This option will set the parity used in communication on the Data Highway to either none or even. The type selected should match the other communication device(s).

Timeout Value. The timeout configuration option determines how long the terminal will wait for a response before it signals a timeout. The value entered will be in tenths of a second with a range of 2 to 40.

Mode. This sets the Mode that the OIL commands will operate in with the Data Highway. In Foreground Mode the Data Highway OIL interface command executes and returns when finished.

In the Background Mode, the Data Highway OIL command initiates execution and gives control back to the OIL program. When the OIL command has completed, the "Busy Bit" in the Communication Status Register is cleared, allowing return processing on that command. As long as the Busy Bit is set, other OIL commands (other than the Data Highway interface commands) can execute. If the program is in background mode and a second Data Highway interface OIL command is issued before the Busy Bit is cleared, an illegal situation occurs and bit #5 in the Communication Status Register is set.

NOTE

In Background Mode, registers used in the Data Highway command will be changed when the Busy Bit is cleared.

Station Number. This number is used to identify the Terminal as a Station on the Data Highway network. This option will be a three digit octal address in the range of 000 to 377. Refer to the PLC manual for recommended addresses.

NOTE

A different address should be used for each device on the Data Highway.

12.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port (in this case, two 25-pin). The figure below illustrates the configuration of the test plugs.

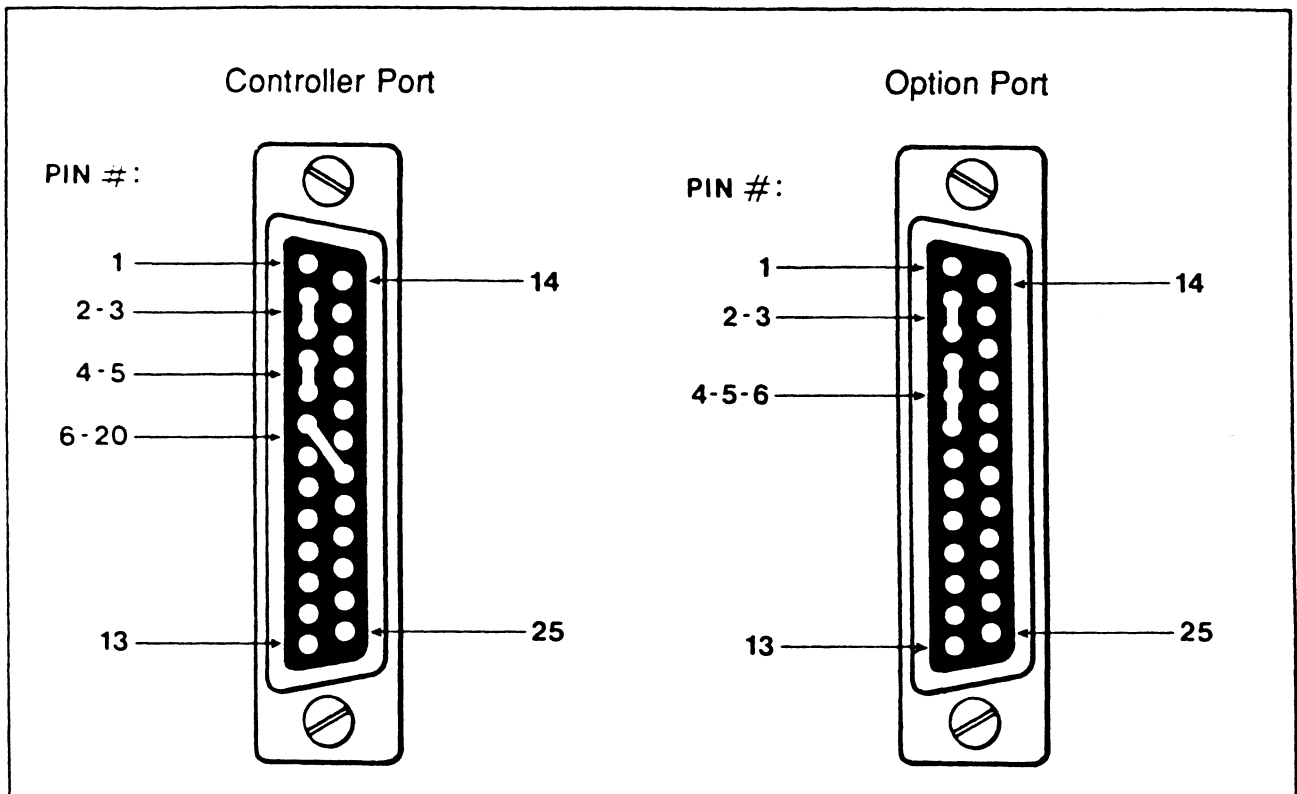


Figure 12-4 Serial Port Test Plugs

12.5 COMMUNICATION STATUS REGISTER

The Communication Status Register contains the status about the message that was just transferred. The Most Significant Byte is the status returned in the message from the communication module (the STS Status Byte). The Least Significant Byte contains information about message transfer status on the terminal end.

If the bit noted is set, the condition indicated is true.

Status Register:

MSB -	message status from Interface Module
LSB -	Bit 0 = Transfer Error
	Bit 1 = Receive Error
	Bit 2 = Timeout
	Bit 3 = Parity Error
	Bit 4 = Busy Bit
	Bit 5 = Cannot Execute Now
	Bits 6 and 7 = NOT USED

Refer to Section 12.3 for explanation of Bits 4 and 5.

12.6 COMMANDS

This section describes the commands that are specific to the 4800-E6 Allen-Bradley Data Highway Interface.

12.6.1 GET DATA FROM DATA TABLE (GET)

Syntax:

```
GET dest,addr,lngth,dreg,sreg
```

where

dest is a 3 digit (octal) Station address of a remote PLC

addr is an octal address in a target Station from which data is to be read

lngth is the number of words (16-bit values) to read, starting at addr

dreg is the first destination register number to store data

sreg is the Communication Status Register

The command GET gets 16-bit data from a specified area of the data table on the PLC. Refer to Section 12.5 for the Communication Status Register information.

Example:

```
GET 110,300,10,#30,#20
```

This command will read 10 words of data starting at address 300 (octal) of the target device. The data returned will be stuffed into registers 30 through 39. Register 20 will contain the Communication Status.

12.6.2 GET DIAGNOSTIC STATUS (GETDIAG)

Syntax:

GETDIAG dest,dreg,sreg

where

dest is a 3 digit (octal) Station address of a remote PLC

dreg is the first destination register number to store data

sreg is the Communication Status Register

The command GETDIAG reads 10 bytes of Diagnostic Status from a remote PLC device, or Station Interface Module. Refer to Section 12.5 for the Communication Status Register information.

Example:

GETDIAG 110,#30,#20

This command will read 10 bytes of status into consecutive 16-bit registers 30 through 39. Register 20 will contain the Communication Status.

12.6.3 GET A BIT OF DATA FROM DATA TABLE (GETIO)

Syntax:

GETIO dest,addr,bit,dreg,sreg

where

dest is a 3 digit (octal) Station address of a remote PLC

addr is an octal address in a target Station from which data is to be read

bit is the bit number desired in the word being read (values 0 - 7)

dreg is the first destination register number to store data

sreg is the Communication Status Register

This command reads a bit of data from the device data table. Refer to Section 12.5 for the Communication Status Register information.

Example:

GETIO 110,47,4,#30,#20

This command will read 4 bit at byte address 47 (octal) of the target device. On return, register 30 will contain either a TRUE or FALSE value corresponding to the bit read. Register 20 will contain the Communication Status.

12.6.4 WRITE DATA TO A DATA TABLE (PUT)

Syntax:

PUT dest,addr,lngth,creg,sreg

where

dest is a 3 digit (octal) Station address of a remote PLC

addr is an octal address in a target Station from which data is to be read

lngth is the number of words (16-bit values) to read, starting at addr

creg is the first source register number to send data from

sreg is the Communication Status Register

The command PUT writes 16-bit data to the data table on the PLC. Refer to Section 12.5 for the Communication Status Register information.

Example:

GET 110,330,110,#30,#20

This command will read 110 words of data to address 300 (octal) of the target device. The data sent will be copied from terminal registers 30 through 139. Register 20 will contain the Communication Status.

12.6.5 GET A BIT OF DATA TO AN ADDRESS (PUTIO)

Syntax:

PUTIO dest,addr,bit,creg,sreg

where

dest is a 3 digit (octal) Station address of a remote PLC

addr is an octal address in a target Station from which data is to be read

bit is the bit number desired in the word being read (values 0 - 7)

creg is the first source register number to send data from

sreg is the Communication Status Register

This command writes a bit of data to a byte address in the data table. Refer to Section 12.5 for the Communication Status Register information.

Example:

PUTIO 110,204,7,#30,#20

This command will read bit 7 at byte address 204 (octal) of the target device. The value of register 30 will determine whether a TRUE or FALSE value will be written. Register 20 will contain the Communication Status.

Chapter 13

4800-E7: MODICON MODBUS NETWORK

13.1 INTRODUCTION

This chapter contains all of the information that is specific to the 4800-E7 Modicon MODBUS PLC network interface. The protocol to be used by the module to communicate with the MODBUS network is a Full-duplex, asynchronous, point-to-point protocol that closely conforms to the ANSI X3.28 standard.

The specific commands that allow communications are listed in Section 13.6. There they are discussed in detail, including examples. These commands are:

GETIR
GETIS
GETOR
GETOS
PUTRM
PUTRS

13.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC and/or network through a cable. This is not a standard cable. The connectors must be fabricated. Figures 13-1 and 13-2 illustrate the connections that must be made between the Expansion Card port and the PLC. The distance between the two devices must be limited to 50 feet or less.

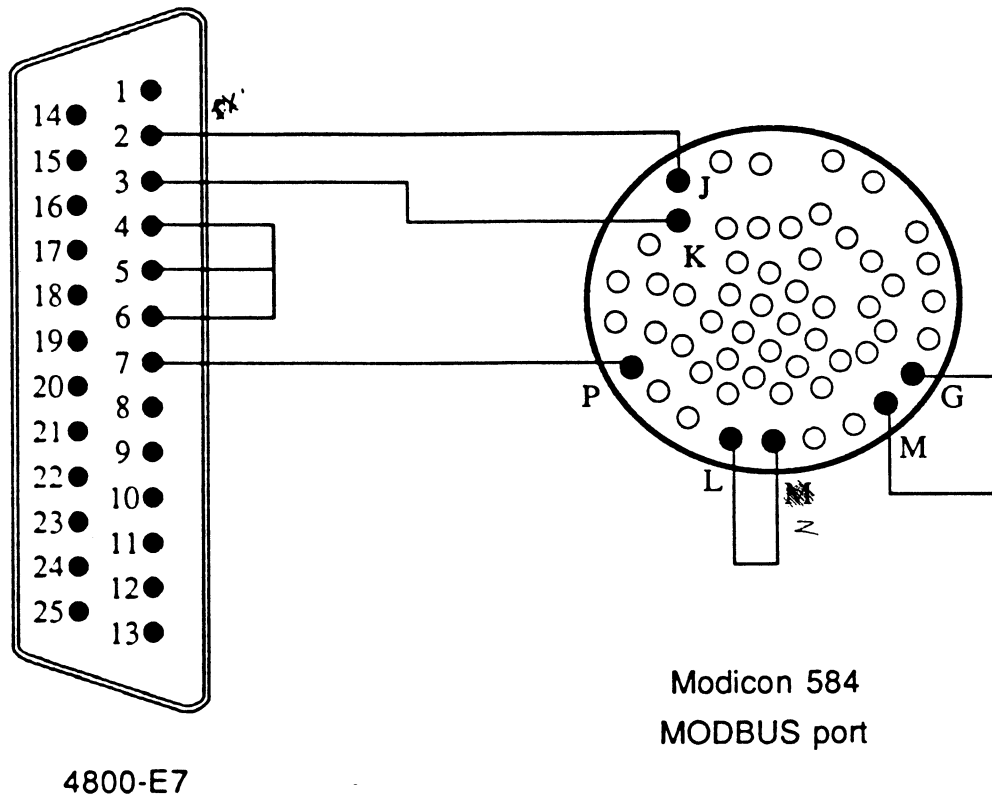
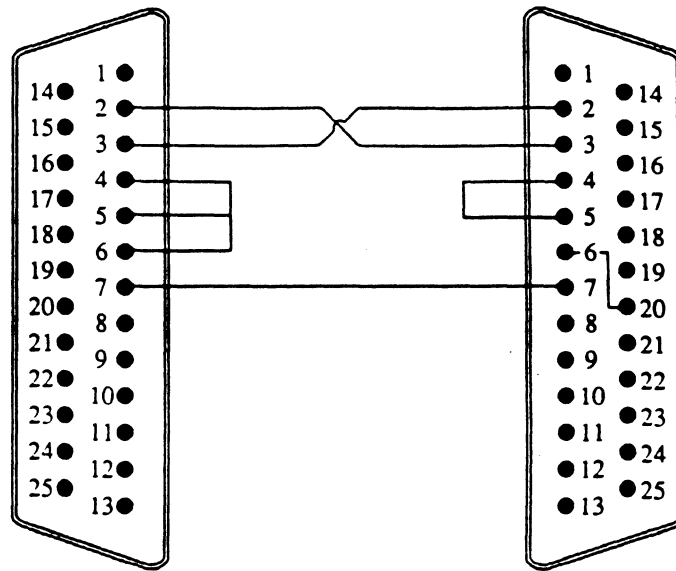


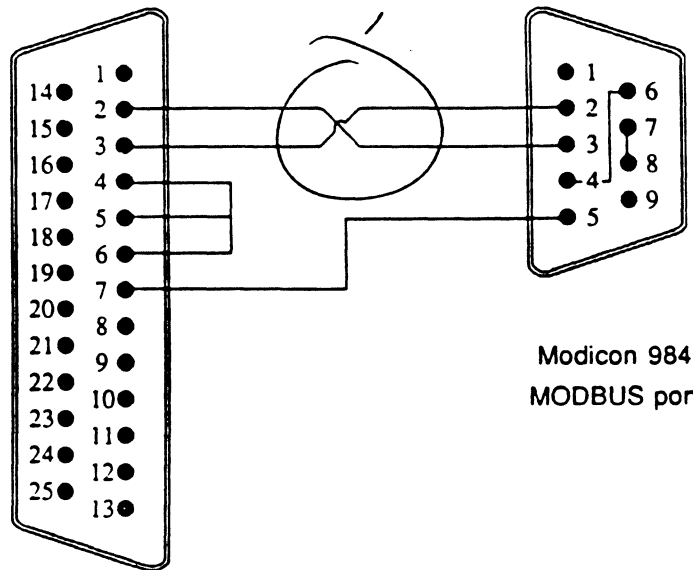
Figure 13-1 Connection to Modicon 584



4800-E7 Module

Modicon 984
MODBUS port

*should be
pin 2 to pin 2 and pin 3 to pin 3*



4800-E7 Module

Modicon 984
MODBUS port

Figure 13-2 Connection to Modicon 984

13.3 CONFIGURATION MENU

The configuration menus are called up from the Main Menu (discussed in Chapter 3). The MODBUS Menu looks like the following:

```

-- MODBUS Port Configuration --

6 Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600 7=19.2
0 Parity - 0=even 1=odd
0 1=Parity Enabled          0=Disabled
0 1=RTU                    0=ASCII
0 1=2 Stop Bits            0=1 Stop Bit

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

Figure 13-3 Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Parity. There are two parity options. The first will set the parity used in communication on the MODBUS network to either odd or even. The type selected should match the other communication device(s). The second will enable or disable the parity.

RTU/ASCII. This option selects the way that the terminal will communicate with the MODBUS, either RTU or ASCII.

Stop Bits. This sets how many Stop Bits will be used in communications with the PLC to either one or two. The number selected should match the other communication device(s).

13.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plugs.

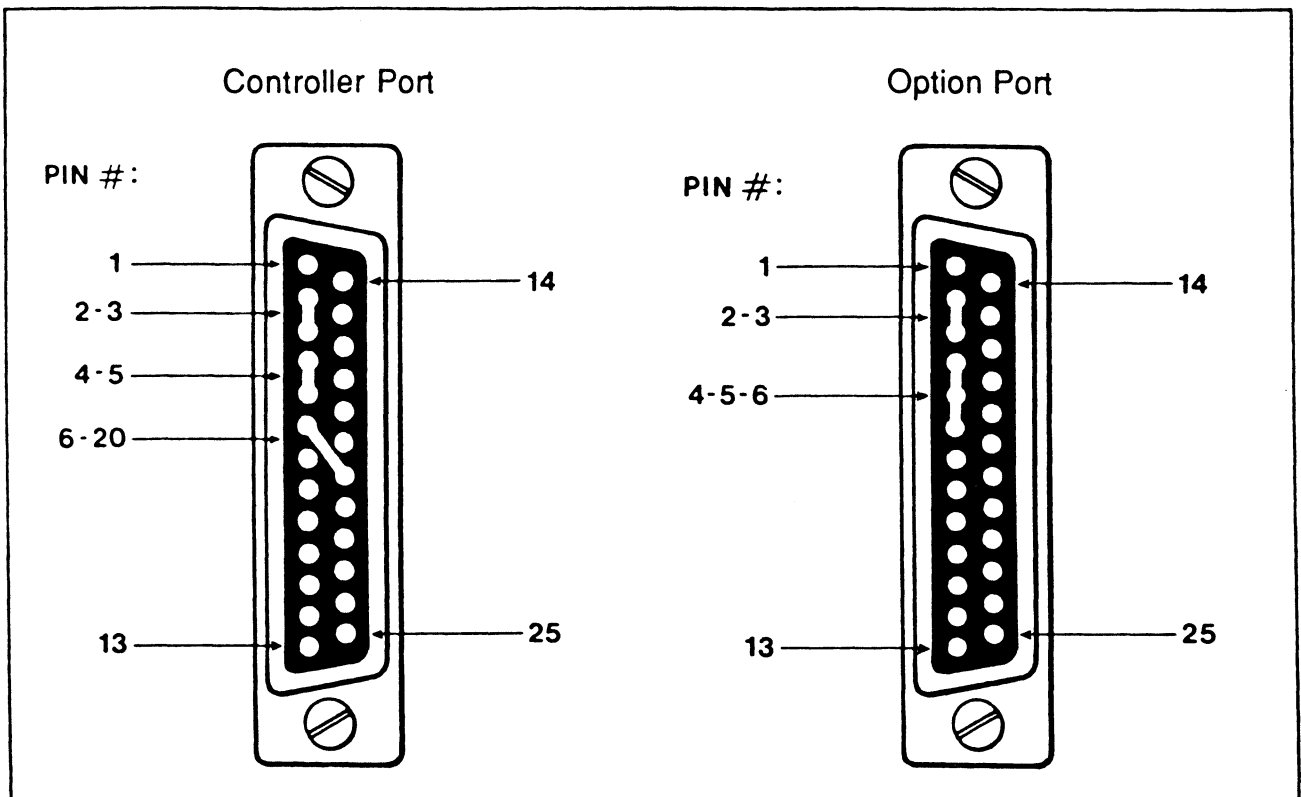


Figure 13-4 Serial Port Test Plugs

13.5 COMMUNICATION STATUS REGISTER

The Communication Status Register contains the status about the message that was just transferred. The Most Significant Byte is the status returned in the message from the communication module. The Least Significant Byte contains information about message transfer status on the terminal end.

If the bit noted is set, the condition indicated is true.

Status Register:

MSB -	message status from Interface Module
LSB -	Bits 0 - 2 = MODBUS exception codes*
	Bit 3 = Transfer Error
	Bit 4 = Receive Error
	Bits 5 and 6 = NOT USED
	Bit 7 = Timeout

- * Refer to the MODBUS user manual for an explanation of the exception codes.

13.6 COMMANDS

This section describes the commands that are specific to the 4800-E6 Allen-Bradley Data Highway Interface.

13.6.1 GET INPUT REGISTER (GETIR) (From MODBUS Slave)

Syntax:

GETIR addr,streg,nreg,dreg,sreg

where

addr is an address in the MODBUS slave from which data is to be read (1 to 247)

streg is the starting input register to read. It is not necessary to specify input registers as lxxx since the command assumes the "I" prefix.

nreg is the number of registers to read

dreg is the first destination register number to store data

sreg is the Communication Status Register

The command GETIR allows OIL to read the contents of input registers in the PLC. This command is analogous to the MODBUS function 04, Read Input Register. Refer to Section 13.5 for the Communication Status Register information.

Example:

GETIR 1,9,1,#100,#11

This command will read input register 3009 from slave PLC number 1. The data returned will be stuffed into register 100. Register 11 will contain the Communication Status.

13.6.2 GET INPUT STATUS (GETIS) (From MODBUS Slave)

Syntax:

GETIS addr,spoint,npoint,dreg,sreg

where

addr is an address in the MODBUS slave from which data is to be read (1 to 247)

spoint is the starting input point to read. It is not necessary to specify input registers as 3xxx since the command assumes the "3" prefix.

npoint is the number of points to read

dreg is the first destination register number to store data

sreg is the Communication Status Register

The command GETIS allows OIL to read the status of input coils in the PLC. The data returned is placed into consecutive data registers low byte first, so that a single byte or an odd byte appears in the low end of that register. This command is analogous to the MODBUS function 02, Read Input Status. Refer to Section 13.5 for the Communication Status Register information.

Example:

TR 2,#40
GETIS #40,197,22,#25,#12

This command will read input coils 1197 to 1218 from slave PLC number 2. The data returned will be stuffed into registers 25 and 26. Register 12 will contain the Communication Status.

13.6.3 GET OUTPUT REGISTER (GETOR) (From MODBUS Slave)

Syntax:

GETOR addr,streg,nreg,dreg,sreg

where

addr is an address in the MODBUS slave from which data is to be read (1 to 247)

streg is the starting output register to be read. It is not necessary to specify output registers as ~~4xxx~~ since the command assumes the "4" prefix.

nreg is the number of registers to read
^{4xxxx}

dreg is the first destination register number to store data

sreg is the Communication Status Register

The command GETIR allows OIL to read the contents of output registers in the PLC. This command is analogous to the MODBUS function 03, Read Output Register. Refer to Section 13.5 for the Communication Status Register information.

Example:

GETOR 2,108,3,#100,#11

This command will read output or holding registers 4108 to 4110 from slave PLC number 2. The data returned will be stuffed into registers 100 through 102. Register 11 will contain the Communication Status.

13.6.4 GET OUTPUT STATUS (GETOS) (From MODBUS Slave)

Syntax:

GETOS addr,spoint,npoint,dreg,sreg

where

addr is an address in the MODBUS slave from which data is to be read (1 to 247)

spoint is the starting input point to read. It is not necessary to specify input registers as 0xxx since the command assumes the "0" prefix.

npoint is the number of points to read

dreg is the first destination register number to store data

sreg is the Communication Status Register

The command GETOS allows OIL to read the status of output coils in the PLC. The data returned is placed into consecutive data registers low byte first, so that a single byte or an odd byte appears in the low end of that register. This command is analogous to the MODBUS function 01, Read Output Status. Refer to Section 13.5 for the Communication Status Register information.

Example:

GETOS 2,20,37,*25,*12

This command will read output coils 0020 to 0056 from slave PLC number 2. The data returned will be stuffed into registers 25 through 27. Bit 0 of data register 25 contains the status of coil 20, while bit 15 of the register contains the status of coil 35. Bit 0 of data register 26 contains the status of coil 36, while bit 15 of the register contains the status of coil 51. Bit 0 of data register 27 contains the status of coil 52, while bit 4 of the register contains the status of coil 56. Bits 5-15 of data register 27 are 0. Register 12 will contain the Communication Status.

13.6.5 **MODIFY MULTIPLE REGISTERS (PUTRM) (To MODBUS Slave)**

Syntax:

PUTRM addr,streg,nreg,dreg,sreg

where

addr is an address in the MODBUS slave from which data is to be read (1 to 247)

streg is the starting output register to be modified. It is not necessary to specify output registers as 4xxx since the command assumes the "4" prefix.

nreg is the number of registers to modify

dreg is the first destination register number to store data

sreg is the Communication Status Register

The command PUTRM allows OIL to modify the contents of a series of output or holding registers in the PLC. This command is analogous to the MODBUS function 16, Preset Multiple Registers. Refer to Section 13.5 for the Communication Status Register information.

Example:

```
TR 100,#20
TR 101,#21
TR 102,#22
PUTRM 2,136,3,#20,#12
```

This command will modify the contents of holding registers 4136, 4137, and 4138 in slave PLC number 2 to 100, 101, and 102, respectively. Register 12 will contain the Communication Status.

13.6.6 **MODIFY A SINGLE REGISTER (PUTRS) (To MODBUS Slave)**

Syntax:

PUTRS addr,reg,data,sreg

where

addr is an address in the MODBUS slave from which data is to be read (1 to 247)

reg is the output register to be modified. It is not necessary to specify the output register as 4xxx since the command assumes the "4" prefix.

data is the value to place into the register

sreg is the Communication Status Register

The command PUTRS allows OIL to modify the contents of a single output or holding register in the PLC. This command is analogous to the MODBUS function 06, Preset Single Register. Refer to Section 13.5 for the Communication Status Register information.

Example:

PUTRS 2,136,926,#12

This command will modify the contents of holding register 4136 in slave PLC number 2 to 926. Register 12 will contain the Communication Status.

Chapter 14

4800-E8: TEXAS INSTRUMENTS SERIES 500

14.1 INTRODUCTION

This chapter contains all of the information that is specific to the 4800-E8: Texas Instruments Series 500 PLC interface. The protocol to be used by the module to communicate with the Texas Instruments PLC is the Non-Intelligent Data Link Protocol using Series 500 Communication Task Codes.

The specific commands that allow communications are listed in Section 14.6. There they are discussed in detail, including examples. These commands are:

GET
GETIO
GETR
PUT
PUTIO
PUTR

14.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC through a cable. This is not a standard cable. The connectors must be fabricated. Figure 14-1 below illustrates the connections that must be made between the Expansion Card port and the PLC.

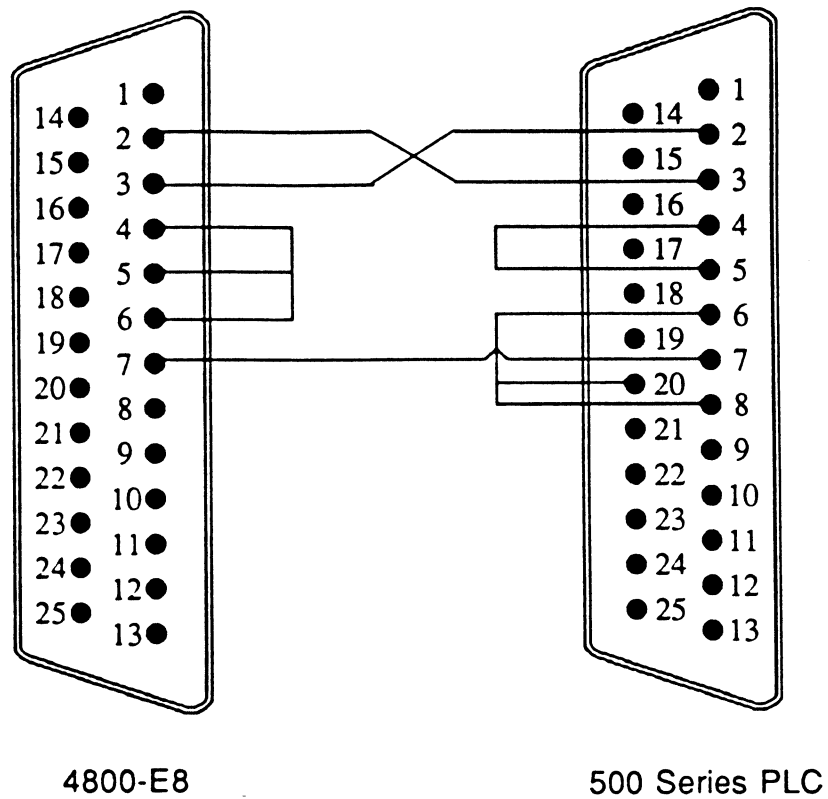


Figure 14-1 Connections

14.3 CONFIGURATION MENU

The configuration menus are called up from the Main Menu (discussed in Chapter 3). The Texas Instruments PLC Port Menu looks like the following:

```
-- Texas Instruments PLC Port Configuration Menu --  
6 Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600 7=19.2  
0 Parity - 0=even 1=odd  
0 1=Parity Enabled          0=Disabled  
0 1=8 Data Bits             0=7 Data Bits  
0 1=2 Stop Bits             0=1 Stop Bit  
  
Use <UP ARROW>, <DOWN ARROW>, <LEFT ARROW>, <RIGHT ARROW>.  
Use values 0 through 9.  
"C" for next configuration menu, <RET> or <ENTER> to quit.
```

Figure 14-2 Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Parity. There are two parity options. The first will set the parity used in communication on the TI-way network to either odd or even. The type selected should match the other communication device(s). The second will enable or disable the parity.

Data Bits. This sets how many Data Bits will be used in communications with the PLC to either seven or eight. The number selected should match the other communication device(s).

Stop Bits. This sets how many Stop Bits will be used in communications with the PLC to either one or two. The number selected should match the other communication device(s).

14.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port (in this case, two 25-pin). The figure below illustrates the configuration of the test plugs.

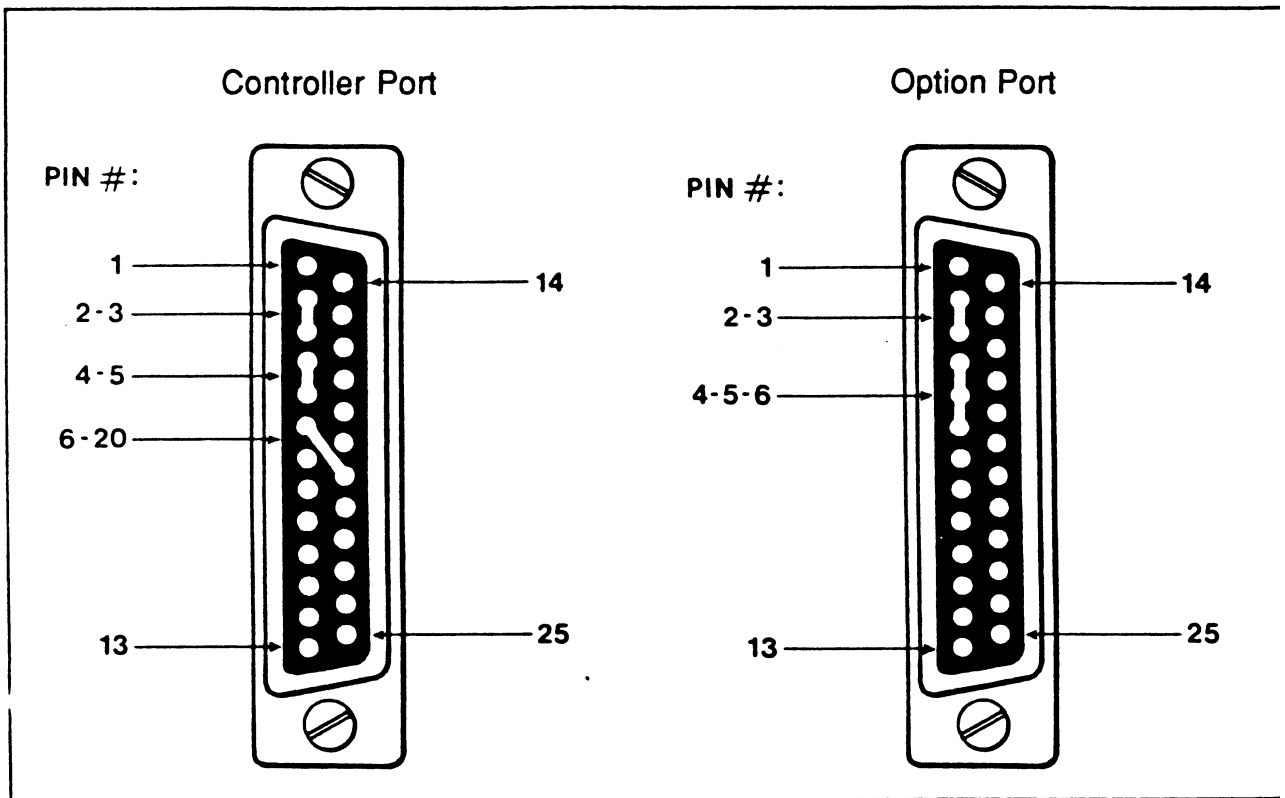


Figure 14-3 Serial Port Test Plugs

14.5 COMMUNICATION STATUS REGISTER

The Communication Status Register contains the status about the message that was just transferred. The upper byte contains the error status returned. When an error is indicated, the lower byte will contain a value indicating the specific error type. A zero value in the entire register indicates a successful transfer.

If the bit noted in the upper byte is set, the condition indicated by the lower byte value is true.

Status Register:

Bit 15 = Timeout
Bit 14 = Receive Error
Bit 13 = Transfer Error *
Bit 12 = Task Error **

* The Transfer Error Code values are:

02 = address out of range
03 = requested data not found
04 = illegal task code request
05 = request exceeds program memory size
07 = fatal error detected
08 = keylock protect error
09 = incorrect amount of data sent with request
0A = illegal request in current PLC mode
0D = odd number of ASCII characters received
0E = illegal write to non-volatile program memory
0F = data not inserted
10 = invalid data sent with command
11 = invalid operation in current PLC mode

** The Task Error Code values are:

01 = character parity framing or overrun error
02 = non-HEX/ASCII character received
03 = invalid character count field
04 = character count error
05 = checksum error

14.6 COMMANDS

This section describes the commands that are specific to the 4800-E8 Texas Instruments Series 500 interface.

14.6.1 GET PLC DATA (GET)

Syntax:

GET device,num,dreg,sreg

where

device is the device type:

- 0 - input memory (WX)
- 1 - output memory (WY)
- 2 - variable memory (V)
- 3 - constant memory (K)
- 4 - drum preset counter (DPC)
- 5 - drum timer current count (DTC)
- 6 - drum timer base (DTB)
- 7 - timer/counter preset count (TCP)
- 8 - timer/counter count count (TCC)

num is the control device number or address

dreg is the destination register number to store data

sreg is the Communication Status Register

The command GET reads a single 16-bit word, register, or control device parameter from a PLC and places the result in a specified register. Refer to Section 14.5 for the Communication Status Register information.

All address start at 1, and increment by 1 for each additional element. The only unusual address form is for drum preset counters (DPCs), which require both a drum number and step values. This is gained by multiplying the drum number by 256 and adding the result to the drum step. For example, drum number 1, step 2 would be calculated: $(1 \times 256) + 2 = 258$.

Example:

GET 2,100,#24,#23

This command will read register 100. The data returned will be stuffed into register 24. Register 23 will contain the Communication Status.

14.6.2 GET PLC I/O POINT (GETIO)

Syntax:

GETIO type,addr,dreg,sreg

where

type is the point type:

- 0 - discrete IR input (X)
- 1 - discrete IR output (Y)
- 2 - discrete control point (C)

addr is the discrete address

dreg is the destination register number to store data

sreg is the Communication Status Register

The command GET reads a discrete I/O point. A "0" is returned if it is off, while a "1" is returned if on. Refer to Section 14.5 for the Communication Status Register information.

Example:

GETIO 2,16,#24,#23

This command will read control point 16. The value returned will be stuffed into register 24. Register 23 will contain the Communication Status.

14.6.3 GET PLC REGISTER BLOCK (GETR)

Syntax:

GETR addr,num,dreg,sreg

where

addr is the beginning variable number or address

num is the number of variables to read

dreg is the destination register number to store data

sreg is the Communication Status Register -

The command GET reads a group of 16-bit variables from a PLC and places the result in a specified register. A maximum of 128 registers may be transferred in a single command. Refer to Section 14.5 for the Communication Status Register information.

Example:

GET 100,6,#24,#23

This command will read PLC variable memory locations 100 through 105. The data returned will be stuffed into registers 24 through 29. Register 23 will contain the Communication Status.

14.6.4 PUT PLC DATA (PUT)

Syntax:

PUT device,num,dreg,sreg

where

device is the device type:

- 0 - input memory (WX)
- 1 - output memory (WY)
- 2 - variable memory (V)
- 3 - constant memory (K)
- 4 - drum preset counter (DPC)
- 5 - drum timer current count (DTC)
- 6 - drum timer base (DTB)
- 7 - timer/counter preset count (TCP)
- 8 - timer/counter count count (TCC)

num is the control device number or address

dreg is the source register number

sreg is the Communication Status Register

The command PUT writes a single 16-bit terminal register to a specific PLC memory location, register, or controller device parameter. Refer to Section 14.5 for the Communication Status Register information.

All address start at 1, and increment by 1 for each additional element. The only unusual address form is for drum preset counters (DPCs), which require both a drum number and step values. This is gained by multiplying the drum number by 256 and adding the result to the drum step. For example, drum number 1, step 2 would be calculated: $(1 \times 256) + 2 = 258$.

Example:

PUT 2,100,#24,#23

This command will write the contents of register 24 out to PLC variable memory location 100. Register 23 will contain the Communication Status.

14.6.5 PUT PLC I/O POINT (PUTIO)

Syntax:

PUTIO type,addr,dreg,sreg

where

type is the point type:

0 - not allowed in PUTIO

1 - not allowed in PUTIO

2 - discrete control point (C)

addr is the discrete address

dreg is the source register number

sreg is the Communication Status Register

The command GET sets a discrete control point to either a "0" or "1". Refer to Section 14.5 for the Communication Status Register information.

Example:

PUTIO 2,12,#22,#23

This command will write the contents of OIL register 22 (must be 1 or 0), to control point 14. Register 23 will contain the Communication Status.

14.6.6 PUT PLC REGISTER BLOCK (PUTR)

Syntax:
PUTR addr,num,dreg,sreg

where

addr is the beginning variable number or address

num is the number of variables to write

dreg is the source register number

sreg is the Communication Status Register

The command PUTR writes a group of 16-bit variables from an OIL register(s) to a block of PLC variable memory. A maximum of 128 registers may be transferred in a single command. Refer to Section 14.5 for the Communication Status Register information.

Example:

PUTR 100,6,#24,#23

This command will write the data from OIL registers 24 through 29 to PLC variable memory locations 100 through 105. Register 23 will contain the Communication Status.

Chapter 15

4800-E9 Eagle Signal PLC Network

15.1 INTRODUCTION

The 4800-E9 package allows the user to read and write specified memory locations within the Eagle Signal EPTAK Programmable Logic Controller via the ECOM Network. The memory locations referenced represent timers, counters, variables, inputs, and outputs. The 4800-E9 module uses Non-Intelligent Data Link Protocol to communicate with the Eagle Signal PLC.

Eagle Signal's "Introduction to ECOM", publication number 5005-780, presents general information about the ECOM Network. It defines commonly used terms and basic rules governing network construction and addressing of controllers.

The specific commands that allow communications are listed in Section 15.6 of this manual, where they are discussed in detail with examples. These commands are:

GET
GETIO
GETR
PUT
PUTR

15.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC through an RS-422 cable. You can order the necessary cable from Eagle (part number CP9200-200) or fabricate the connectors to make your own cable. Figure 15-1 below illustrates the connections that must be made between the Expansion Card port and the PLC.

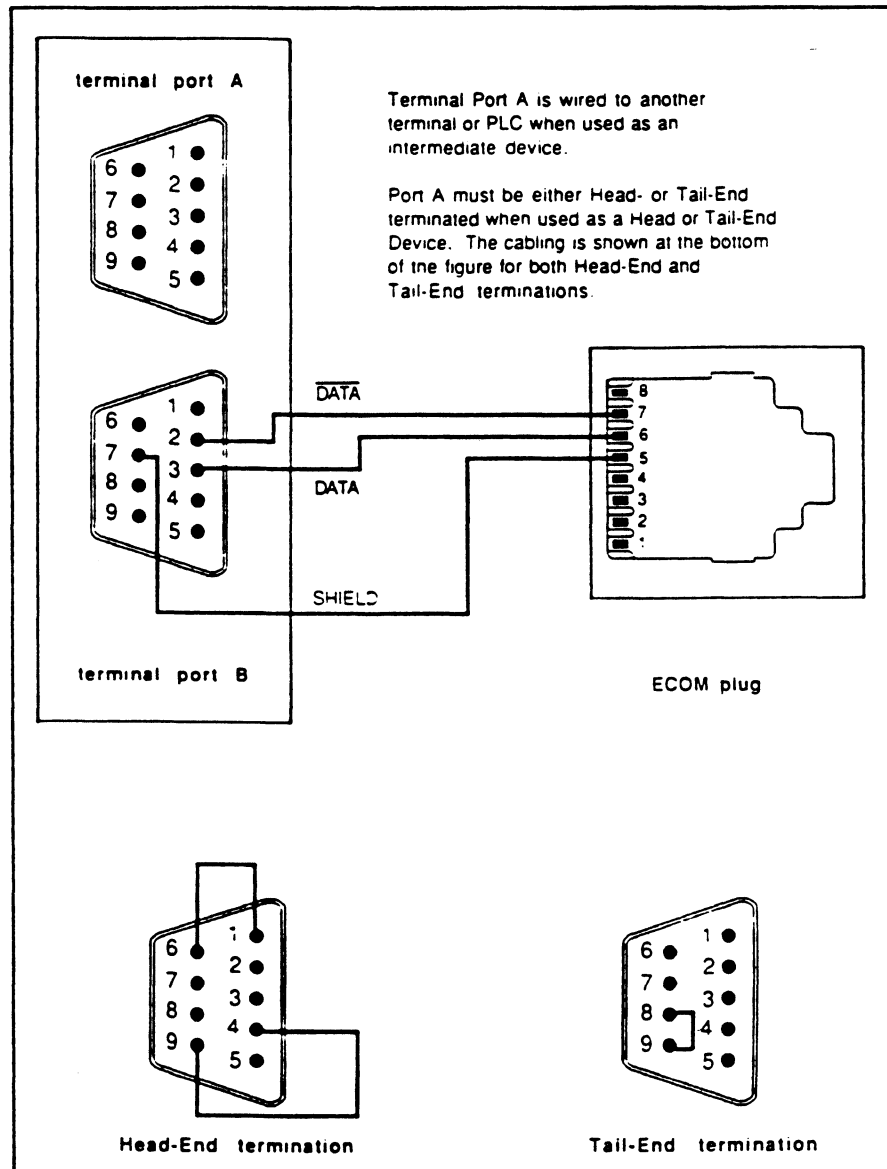


Figure 15-1. Connections

15.3 CONFIGURATION MENU

The configuration menus are called up from the Main Menu (discussed in Chapter 3). Figure 15-2 shows the ECOM Port Configuration Menu.

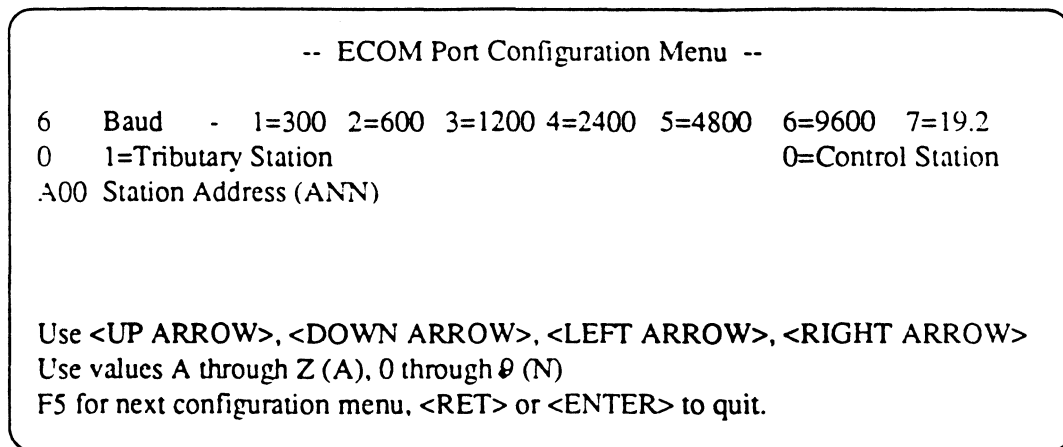


Figure 15-2. Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Tributary/Control Station. This menu item will designate the terminal as either a master or a slave on the ECOM Network.

Station Address. The station address is the label assigned to the terminal, consisting of a letter and a two digit decimal number. If the terminal is a Control Station, the number is 00 (the letter should match that of the PLC to which the terminal is connected). If the terminal is a Tributary Station, the letter matches other devices on the secondary link, and the number identifies the terminal.

15.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plugs.

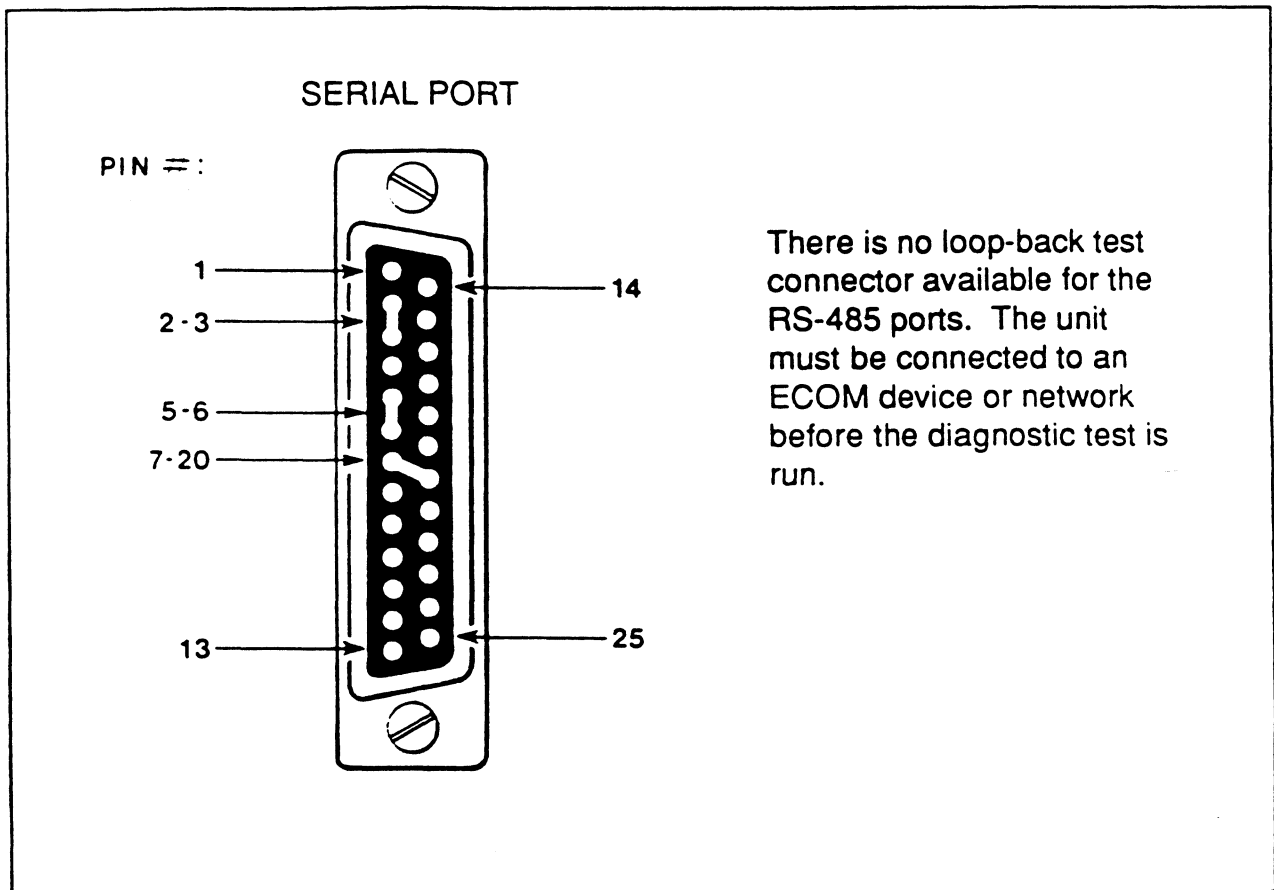


Figure 15-3. Serial Port Test Plugs

15.5 COMMUNICATION STATUS REGISTER

The Communication Status Register contains the status about the message that was just transferred. When the register is zero, no errors occurred. Errors are indicated by individual bits in the Status Register.

If the bit noted is set, the condition indicated is true.

NOTE

Two different error conditions can be flagged by bit 6, dependent on information in bit 7. If bit 7 is set, bit 6 indicates a tributary timeout. If bit 7 is NOT set, bit 6 indicates an OIL parameter error.

- Bit 0 = Select Error
- Bit 1 = Transmit Error
- Bit 2 = Poll Error
- Bit 3 = Receive Error
- Bit 4 = Unable to Execute Command
- Bit 5 = PLC Does Not Respond
- Bit 6 = OIL Parameter Error *or* Tributary Timeout (see NOTE above)
- Bit 7 = Timeout
- Bits 8-15 = NOT USED

15.6 COMMANDS

This section describes the commands that are specific to the 4800-E9 Eagle Signal ECOM Network Interface.

15.6.1 Get PLC Data (GET)

Syntax:

GET "addr",fmlly,dvc,num,dreg,sreg

Where:

addr is a three character address of the source controller; the first character must be an uppercase letter (A-Z) and the last two character must be a number (00-99)

fmlly is the EPTAK controller family

- 1 - 2X5, Micro 190/190+
- 2 - 7000
- 3 - Eagle Series

dvc is the device type (see Table 15-1 on next page) --

num is the control device number

dreg is the destination register number to store data

sreg is the Communication Status Register

The GET command reads a single 16-bit word, register, or control device parameter from a PLC and places it in the specified OIL register. Refer to Section 15.5 for the Communication Status Register information.

Table 15-1. Available Device Types

#	Description	PLC Compatibility			
		2X5	Micro	7000	Eagle
0	DAT register or RAM location	1-256	1-256	1-65000	--
1	timer actual	1-128	1-128	1-255	1-125
2	timer setpoint	1-128	1-128	1-255	1-125
3	counter actual	1-32	1-32	1-255	1-100
4	counter setpoint	1-32	1-32	1-255	1-100
5	PID actual (in 2X5, same as Analog xxx actual)	101-116	--	1-255	1-64
6	PID setpoint (in 2X5, same as Analog xxx setpoint)	101-116	--	1-255	1-64
7	PID gain	101-116	--	1-255	1-64
8	PID rate	101-116	--	1-255	1-64
9	PID reset	101-116	--	1-255	1-64
10	PID output	--	--	1-255	1-64
11	PID cycle time	--	--	--	1-64
12	alarm hi	--	--	1-255	1-64
13	alarm lo	--	--	1-255	1-64
14	chassis analog in	101-116	--	●	●
15	chassis analog out	117-132	--	●	●
16	remote analog in	--	--	■	■
17	remote analog out	--	--	X	X
20	Eagle Series data register	--	--	--	1-999*
21	floating point data register	--	--	--	1-199*
22	control relay	--	--	--	1-999*
23	retentive control relay	--	--	--	1-499*
24	PID manual output in engineering units	--	--	--	1-643
25	PID actual value in engineering units	--	--	--	1-64
26	PID setpoint in engineering units	--	--	--	1-64
27	PID high deviation alarm limit	--	--	--	1-64
28	PID low deviation alarm limit	--	--	--	1-64
99	absolute address	0-65535	0-65535	0-65535	0-65535

(●, ■, X, *): see table notes on following page)

Table Notes:

- (14 & 15) Device numbers are of the form cmmk where:
 - c = chassis number (1-8)
 - mm = module number (03-16)
 - k = circuit number (1-8 for analog inputs, 1-4 for analog outputs)
- (16) Device number are of the form dscc where:
 - d = CP716 driver number (1-8)
 - s = CP2035 station number (1-8)
 - cc = circuit number (01-16)
- I (17) Device numbers are of the form dscc where:
 - d = CP716 driver number (1-8)
 - s = CP2035 station number (1-8)
 - cc = circuit number (17-32)
- (20-23) In the Eagle Series controller, the number of data registers, floating point data registers, control relays and retentive control relays can vary depending on the configuration in the user program. This table shows the initial default settings.

Example:

```
GET "A00",2,1,4,#100,#200
```

This command will read the Timer 4 Actual Value from an EPTAK 7000 at ECOM address "A00", placing the value in Terminal Register 100. The Communication Status goes into register 200.

15.6.2 Get I/O Point (GETIO)

Syntax:

GETIO "addr",fmly,type,x,y,z,dreg,sreg

where:

addr is a three character address of the source controller; the first character must be an uppercase letter (A-Z) and the last two character must be a number (00-99)

fmly is the EPTAK controller family

- 1 - 2X5, Micro 190/190+
- 2 - 7000
- 3 - Eagle Series

type is the I/O type

- 1 - input
- 2 - output

x, y and z are address pointers as detailed in Table 15-2 below

dreg is the destination register number to store data

sreg is the Communication Status Register

Table 15-2. GETIO Address Pointers

controller	x is:	y is:	z is:
2X5, Micro 190/190+	track number (1-4)	always 1	circuit number (1-32)
7000 or Eagle remote	chassis number (1-8)	slot number (3-16)	circuit number (1-16)
7000 or Eagle chassis	driver number (1-8)	station number (1-8)	circuit number (1-32)

The GETIO command reads a discrete I/O point and returns a "0" if it is off, or a "1" if on. Refer to Section 15.5 for the Communication Status Register information.

Example:

GETIO "A01",2,2,1,8,16,#24,#23

This command will read, from an EPTAK 7000 at address "A01," the discrete output in Chassis 1, Slot 8, Circuit 16. If the output is ON, a 1 will be placed in Register 224 of the OIT; if the output is OFF, a 0 is placed in Register 24. The Communication Status goes into Register 23.

15.6.3 Get Register Block (GETR)

Syntax:

GETR "addr",fmly,num,breg,dreg,sreg

where:

addr is a three character address of the source controller; the first character must be an uppercase letter (A-Z) and the last two character must be a number (00-99)

fmly is the EPTAK controller family

- 1 - 2X5, Micro 190/190+
- 2 - 7000
- 3 - Eagle Series

breg is the beginning register number in controller

num is the number of registers to transfer

dreg is the destination register number to store data

sreg is the Communication Status Register

The GETR command reads a group of 16-bit registers from a PLC and places them in the specified terminal data registers. You can transfer a maximum of 128 registers in a single command, but GETR can *only* read consecutive registers. Be aware that in the 2X5 and Micro 190 190+, registers DAT 32 and DAT 33 are NOT consecutive in the controller's memory. Refer to Section 15.5 for the Communication Status Register information.

Example:

GETR "A00",1,100,10,#101,#200

This command will read data registers 100 through 109 (ten registers) from a 2X5 or Micro 190/190+ at ECOM address A00, and place the data in terminal registers 101 through 110. The Communication Status goes in register 200.

15.6.4 Put Data (PUT)

Syntax:

PUT "addr",fmlly,dvc,num,dreg,sreg

where:

addr is a three character address of the source controller; the first character must be an uppercase letter (A-Z) and the last two character must be a number (00-99)

fmlly is the EPTAK controller family

- 1 - 2X5, Micro 190/190+
- 2 - 7000
- 3 - Eagle Series

dvc is the device type (see Table 15-1, page 15-7)

num is the control device number

dreg is the source register number

sreg is the Communication Status Register

The PUT command writes a single 16-bit word, register, or control device parameter to a PLC from the specified terminal data register. Refer to Section 15.5 for the Communication Status Register information.

Example:

PUT "B01".3.20.30.#101.#201

This command will write the value contained in the terminal register 101 into data register 30 of an Eagle Series controller whose ECOM address is B01. The Communication Status is in register 201.

15.6.5 Put Register Block (PUTR)

Syntax:

PUTR "addr",fmly,breg,num,dreg,sreg

where:

addr is a three character address of the source controller; the first character must be an uppercase letter (A-Z) and the last two character must be a number (00-99)

fmly is the EPTAK controller family

- 1 - 2X5, Micro 190/190+
- 2 - 7000
- 3 - Eagle Series

breg is the beginning register number in controller

num is the number of registers to transfer

dreg is the source register number

sreg is the Communication Status Register

The PUTR command writes a group of 16-bit variables to PLC registers from the specified OIL registers. You can transfer a maximum of 128 registers in a single command, but the PUTR command can only write to consecutive registers. Be aware that in the 2X5 and the Micro 190/190+ registers DAT 32 and DAT 33 are not consecutive in the controller's memory. Refer to Section 15.5 for the Communication Status Register information.

Example:

```
PUTR "C02",2,300,25,#100,#199
```

This command will read 25 pairs of 1-byte RAM locations, beginning with register 300, from an EPTAK 7000 whose address is C02. The data will be placed in terminal registers 100 through 124. The Communication Status goes in register 199.

Chapter 16

4800-E10: SQUARE-D SY/MAX NETWORK

16.1 INTRODUCTION

This chapter contains all of the information that is specific to the 4800-E10: Square-D SY/MAX Network interface. The protocol to be used by the module to communicate with the Square-D SY/MAX Network is a full-duplex asynchronous point-to-point protocol that closely conforms to the ANSI X3.28 standard.

The specific commands that allow communications are listed in Section 16.6. There they are discussed in detail, including examples. These commands are:

GET
GETN
GETNN
PUT
PUTN
PUTNN

The 4800-E10 requires a Xycom RS-422 Communication Adapter Module be installed before operation.

16.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC through a cable. This is not a standard cable. The connectors must be fabricated. Figure 16-1 and 16-2 illustrate the connections that must be made between the Expansion Card port and the PLC.

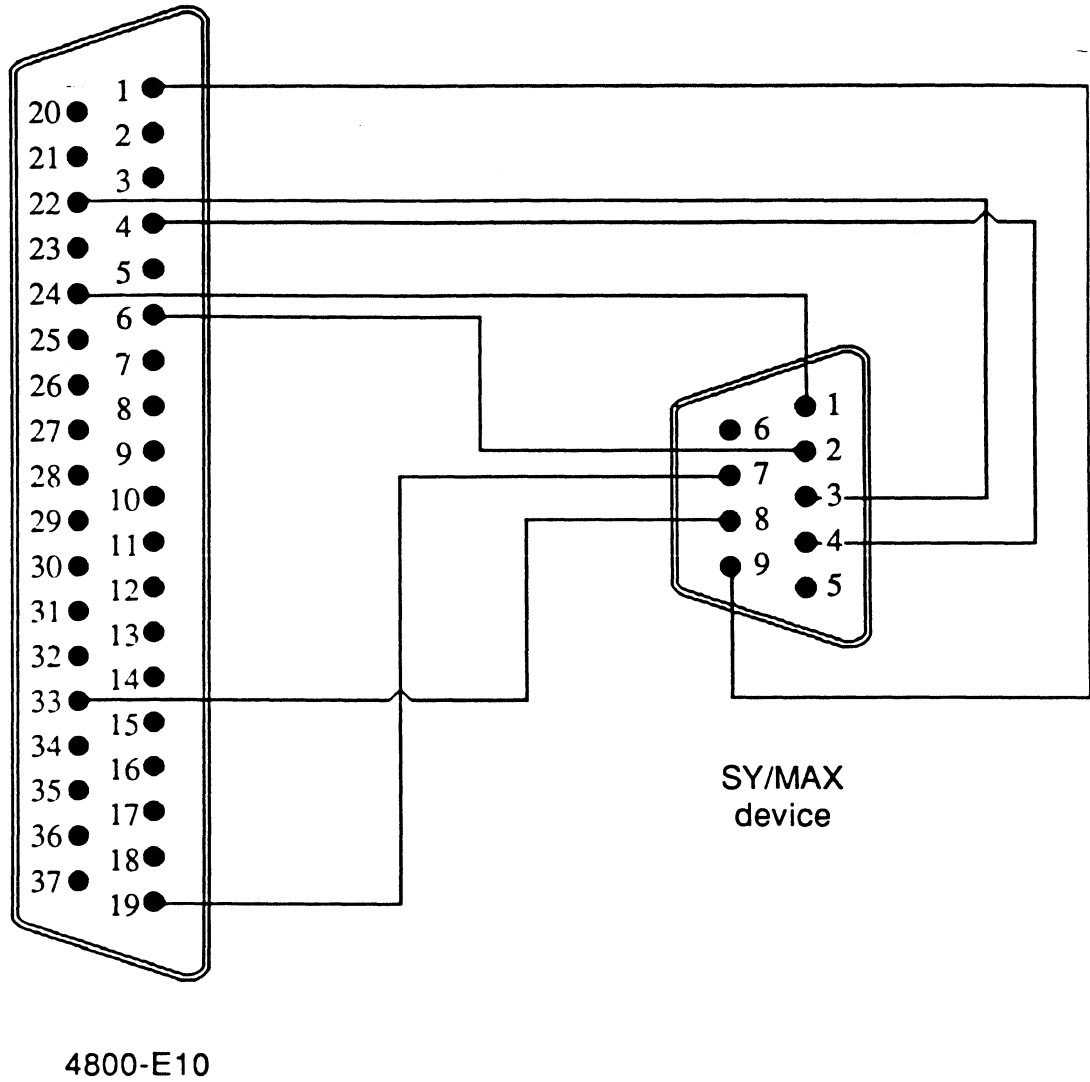


Figure 16-1 Connection to the SY/MAX

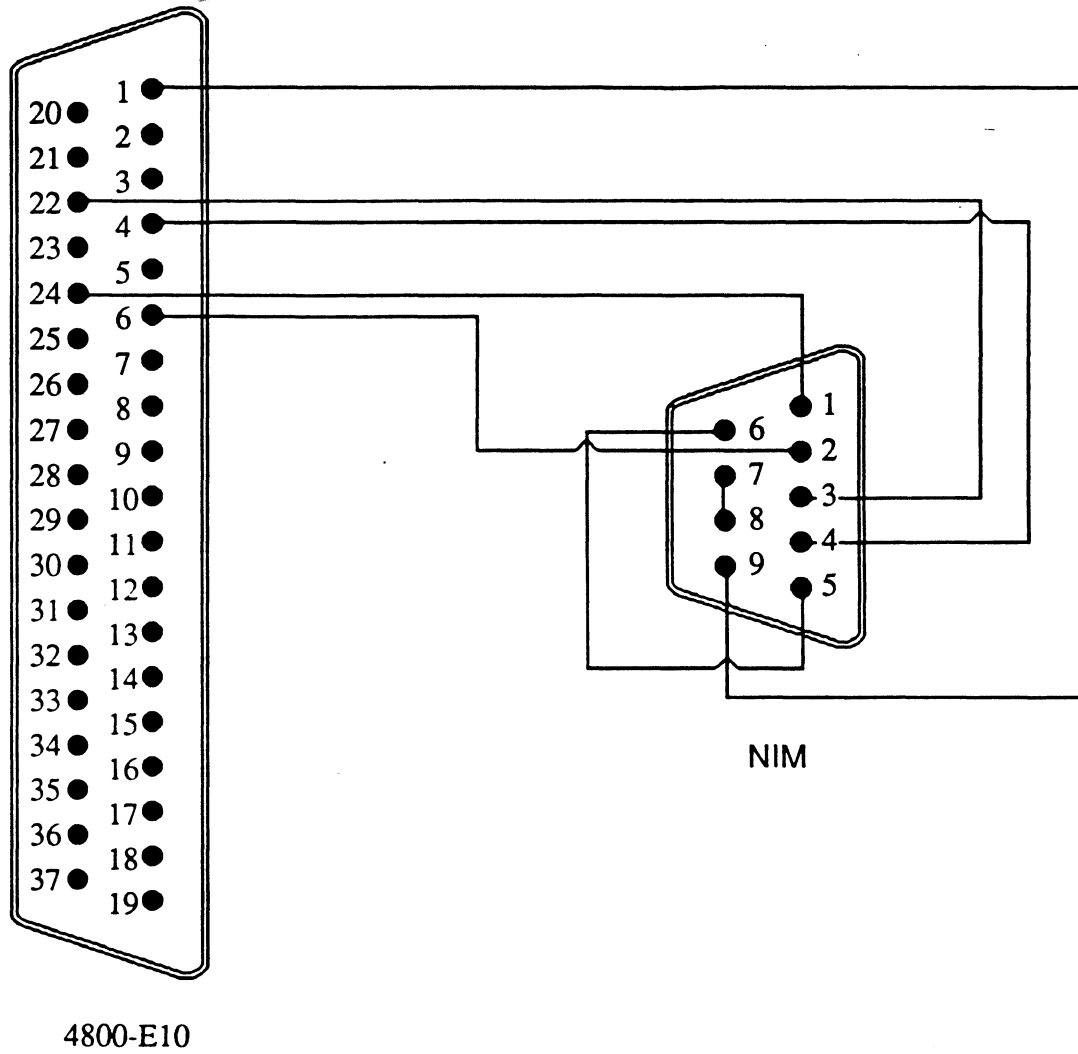


Figure 16-2 Connection to the NIM

16.3 CONFIGURATION MENU

The configuration menus are called up from the Main Menu (discussed in Chapter 3). The SY/MAX Menu looks like the following:

```
-- Square-D Configuration Menu --  
  
6  Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600  
001 Network Interface Module # (000-199)  
01  Response Timeout Value  
  
Use <UP ARROW>, <DOWN ARROW>, <LEFT ARROW>, <RIGHT ARROW>.  
Use values 0 through 9.  
"C" for next configuration menu, <RET> or <ENTER> to quit.
```

Figure 16-3 Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Network Interface Module #. If the terminal is set up in a network configuration, one or more NIMs will be used. The NIM number setting in the Square-D configuration **MUST** be identical to the network device number.

The network device number is determined by two thumbwheel switches on the NIM COM port. The first number (X__) is the number of the COM port being accessed (0 or 1). The second and third (__XX) are the thumbwheel settings (0 to 9 each).

Response Timeout Value. The timeout configuration option determines how long the terminal will wait for a response before it signals a timeout. The value entered will be in seconds, with a range of 1 to 16.

16.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plugs.

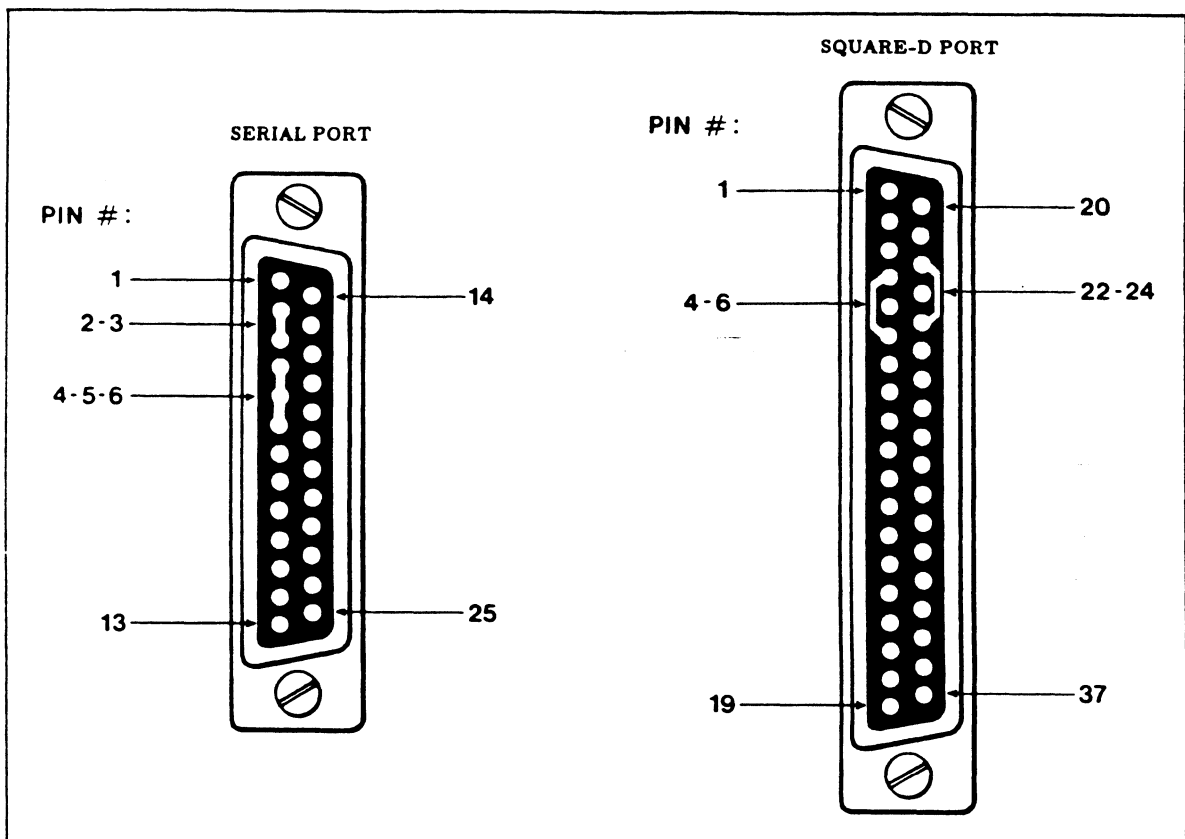


Figure 16-4 Serial Port Test Plugs

16.5 COMMUNICATION STATUS REGISTER

The Communication Status Register in the OIL command contains the status of the message previously transferred. The most significant byte (MSB) is the status returned in the message from the Communication Module. The least significant byte (LSB) contains internal XYCOM information about the transfer status.

If a bit is set in the Status Register then the condition corresponding to that bit described below is true.

Status Register:

MSB: Message status from the SY/MAX module

Bit 15	No Errors
Bit 14	Responding processor in RUN mode
Bit 13	Destination Network Module not found
Bit 12	See PUT command below

Bits 8-11 are different for GET and PUT commands

GET command

Bit 8	Error
Bits 8,10	Illegal Operation - an attempt was made to read from a device with too many registers.
Bits 8,10,11	Receiver Overflow - Too many registers were sent or requested.

PUT command

Bit 8	Error
Bits 8,9	Illegal Address - An attempt was made to write to an address that does not exist.
Bits 8,10	Illegal Operation - An attempt was made to to a device with no registers.
Bits 8,11	Write to a Read Only Register
Bits 8,12	Receiver Overflow - Too many registers were sent to a remote processor.

LSB: Status from XYCOM Communications Board

Bit 0	Transmit Error
Bit 1	Receive Error
Bit 2	Timeout
Bit 3	Parity Error
Bits 4-7	Not Used

16.6 COMMANDS

This section describes the commands that are specific to the 4800-E6 Allen-Bradley Data Highway Interface.

16.6.1 GET REGISTER FROM DESTINATION (GET)

Syntax:

GET addr,length,dreg,sreg

where:

addr is an address in SY/MAX from which data is to be read

length is the number of registers (16-bit values) to read, starting at addr

dreg is the Starting Register number to store received data

sreg is the Communication Status Register

The command GET gets 16-bit data when connected directly to a PLC. Refer to Section 16.5 for Communication Register error bit responses.

Example:

GET 300,10,#30,#20

This command will read 10 registers starting at register 300 of the target device. The data returned will be stuffed into registers 30 through 39. Register 20 will contain the Communication Status.

16.6.2 GET A REGISTER FROM DESTINATION NETWORK DEVICE (GETN)

Syntax:

GETN dest,addr,lngth,dreg,sreg

where:

dest is the address of the network device.

addr is the decimal address in the target device from which data is to be read.

lngth is the number of registers (16 bit values) to read, starting at addr.

dreg is the Starting Register number to store received data into.

sreg is the Communication Status Register.

GETN reads data from a PLC via a Network Interface Module (NIM). Refer to Section 16.5 for Communication Status Register error bit responses.

Example:

GETN 15,20,10,#30,#20

This command will read 10 registers from the destination SY/MAX device at network address 15. The data will be read starting with register 20 of the target device. The data returned will be stuffed into registers 30 through 39. Register #20 will contain the Communication Status.

16.6.3 GET A REGISTER USING NET TO NET (GETNN)

Syntax:

GETNN dest,netn,addr,lngth,dreg,sreg

where:

dest is the address of the network device from which to GET data.

netn is the NET-to-NET routing address.

addr is the decimal address in the target device from which data is to be read.

lngth is the number of registers (16 bit values) to read, starting at addr.

dreg is the starting register number to store data into.

sreg is the Communication Status Register.

This command reads data from a PLC located on another network. Refer to Section 16.5 for Communication Status Register error bit responses.

Example:

GETNN 15,10,20,100,#30,#20

This command will read 100 registers from the destination SY/MAX device following NET-to-NET address 10 at destination address 15. The data will be read starting with register 20 of the target device. The data returned will be stuffed into registers 30 through 139. Register 20 will contain the Communication Status.

16.6.4 WRITE DATA TO A REGISTER (PUT)

Syntax:

PUT addr,lngth,creg,sreg

where:

addr is a register address in SY/MAX into which data is to be written.

lngth is the number of registers (16-bit values) to write, starting at addr.

creg is the Starting Register number to send data from.

sreg is the Communication Status Register.

The PUT command writes 16-bit data to a register in the destination device when connected directly to the PLC. Refer to Section 16.5 for Communication Status Register error bit responses.

Example:

PUT 300,10,#30,#20

This command will write 10 registers starting at register 300 of the target device. The data sent will be from registers 30 through 39. Register 20 will contain the Communication status.

16.6.5 WRITE A REGISTER TO THE DESTINATION DEVICE (PUTN)

Syntax:

PUTN dest,addr,lngth,creg,sreg

where:

dest is the address of the network device to PUT data into.

addr is the decimal address in the target device into which data is to be written.

lngth is the number of registers (16 bit values) to write, starting at addr.

creg is the starting register number to send data from.

sreg is the Communication Status register.

PUTN writes data to a PLC via an NIM. Refer to Section 16.5 for Communication Status Register error bit responses.

Example:

PUTN 15,20,10,#30,#20

This command will write to 10 registers in the destination SY/MAX device starting at register 20. The target device is at network address 15. The data will be written from our registers 30 through 39. Register 20 will contain the Communication Status.

16.6.6 WRITE A REGISTER USING NET TO NET (PUTNN)

Syntax:

PUTNN dest,netn,addr,lngth,creg,sreg

where:

dest is the address of the network device in which to PUT data.

netn is the NET-to-NET routing address.

addr is the decimal address in the target device into which data is to be written.

lngth is the number of registers (16 bit values) to written, starting at addr.

dreg is the starting register number to send data from.

sreg is the Communication Status Register.

This command writes data to a PLC on another network. Refer to Section 16.5 for Communication Status Register error bit responses.

Example:

PUTNN 15,30,20,100,#30,#20

This command will write 100 registers in the destination SY/MAX device starting at register 20. The target device is along the NET-to-NET address of 30, at destination address 15. The data will be written starting from our registers 30 through 139. Register 20 will contain the Communication Status.

Chapter 17

4800-E11: WESTINGHOUSE NUMA-LOGIC NETWORK

17.1 INTRODUCTION

This chapter contains all of the information that is specific to the 4800-E11: Westinghouse Numa-Logic PLC interface. The protocol to be used by the module to communicate with the Westinghouse PLC is specific to the Numa-Logic PLC. Refer to the Westinghouse documentation for details.

The specific commands that allow communications are listed in Section 17.6. There they are discussed in detail, including examples. These commands are:

GET
PUT
PUTIO

17.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC through a cable. This is not a standard cable. The connectors must be fabricated. Figure 17-1 below illustrates the connections that must be made between the Expansion Card port and the PLC.

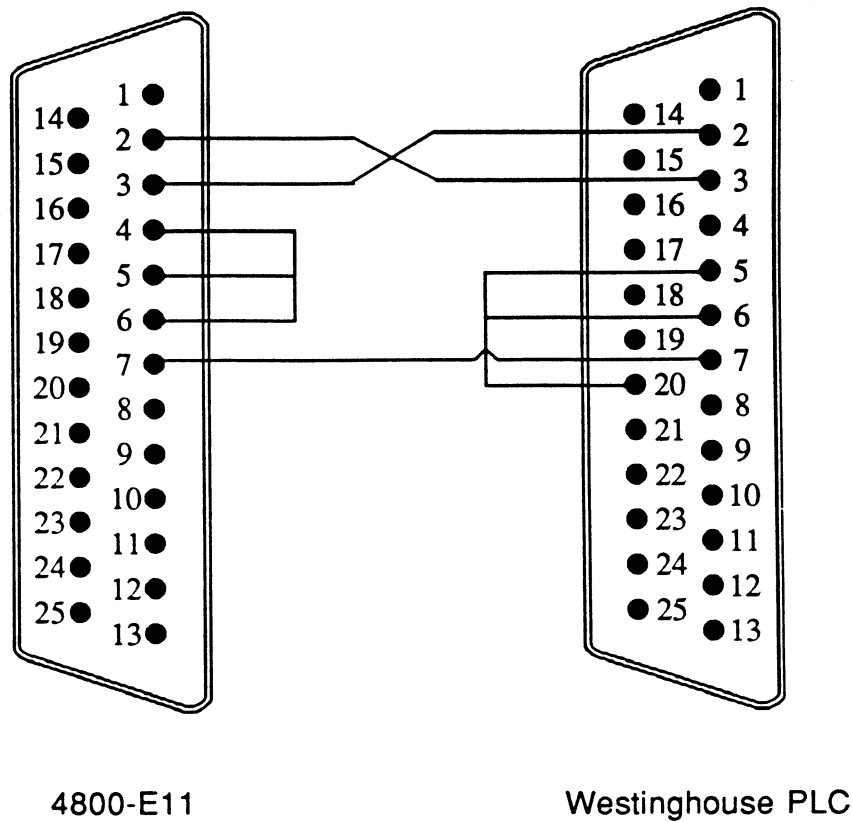


Figure 17-1 Connections

17.3 CONFIGURATION MENU

The configuration menus are called up from the Main Menu (discussed in Chapter 3). The Westinghouse Menu looks like the following:

```
-- Westinghouse Port Configuration --  
  
6 Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600 7=19.2  
2 0=No Parity 1=Even Parity 2=Odd Parity  
1 Timeout Value (1-10 seconds)  
  
Use <UP ARROW>, <DOWN ARROW>, <LEFT ARROW>, <RIGHT ARROW>.  
Use values 0 through 9.  
"C" for next configuration menu, <RET> or <ENTER> to quit.
```

Figure 17-2 Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Parity. This option will set the parity used in communication on the Data Highway to either none or even. The type selected should match the other communication device(s).

Timeout Value. The timeout configuration option determines how long the terminal will wait for a response before it signals a timeout. The value entered will be in seconds, with a range of 1 to 10.

17.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port (in this case, two 25-pin). The figure below illustrates the configuration of the test plugs.

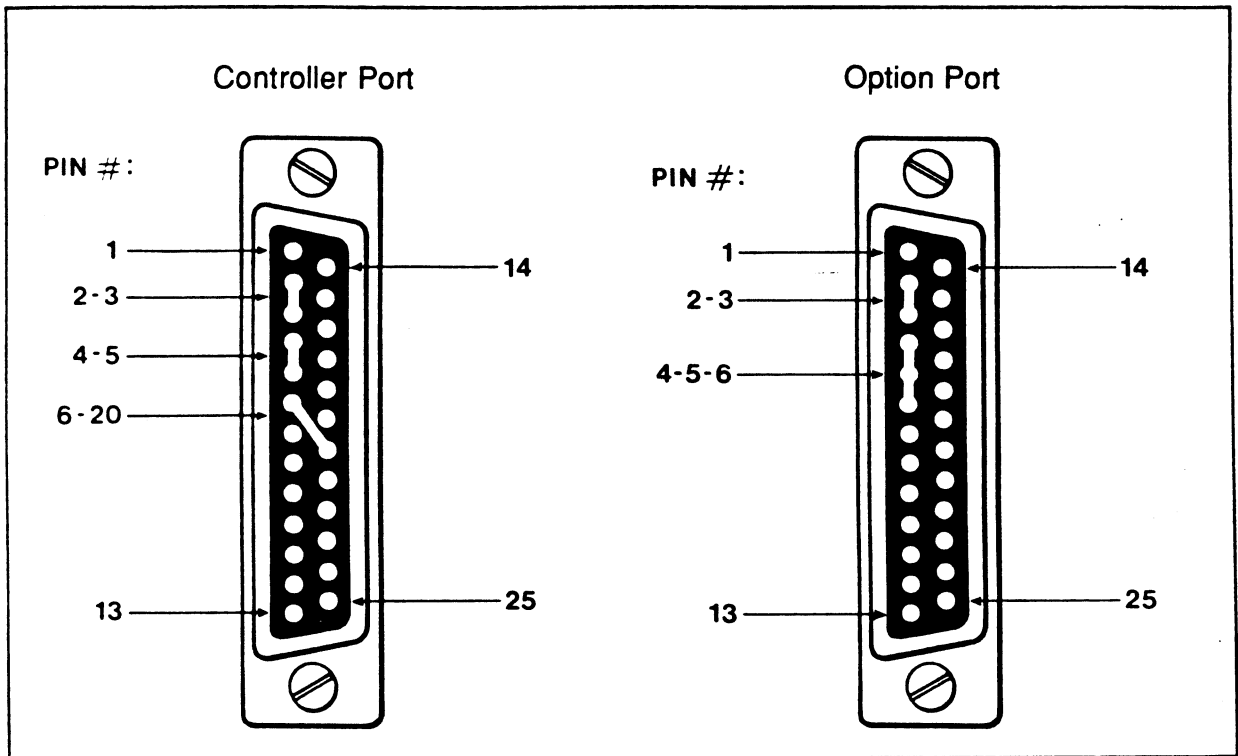


Figure 17-3 Serial Port Test Plugs

17.5 COMMUNICATION STATUS REGISTER

The Communication Status Register contains the status about the message that was just transferred. The Most Significant Byte (MSB) is the status returned from the Westinghouse PLC. The Least Significant Byte (LSB) contains information about the message transfer status. A zero value in the entire register indicates a successful transfer.

If the bit noted is set, the condition indicated is true.

Status Register:

MSB: Bits 8-11 = **NOT USED**
 Bits 12-15 = see below

BIT				
<u>15</u>	<u>14</u>	<u>13</u>	<u>12</u>	
0	0	0	1	= Attempt to Write Ladder While Running
0	0	1	0	= Invalid Command Op. Code
0	0	1	1	= Checksum Error
0	1	0	0	= Command Overrun
0	1	0	1	= Command Aborted
0	1	1	0	= UART Overrun
0	1	1	1	= Invalid Address
1	0	0	0	= UART Framing Error
1	0	0	1	= UART Parity Error

LSB: Bit 0 = Transmit Error
 Bit 1 = Receive Error
 Bit 2 = Timeout
 Bit 3 = Parity Error
 Bit 4 = Synchronization Error
 Bits 5-7 = **NOT USED**

17.6 COMMANDS

This section describes the commands that are specific to the 4800-E6 Allen-Bradley Data Highway Interface.

17.6.1 GET PLC DATA (GET)

Syntax:

```
GET addr,lngth,dreg,sreg
```

where

addr is the address in a target station to read data from

lngth is the number of words (16-bit values) to read, starting at addr

dreg is the destination register number to store data

sreg is the Communication Status Register

The command GET reads a 16-bit word(s) from a data table in the PLC and places the result in a specified register. Refer to Section 17.5 for the Communication Status Register information.

Example:

```
GET &1200,10,#30,#20
```

This command will read 10 words of data starting at address (hex)1200 of the target device. The data returned will be stuffed into registers 30 through 39. Register 20 will contain the Communication Status.

17.6.2 PUT PLC DATA (PUT)

Syntax:

PUT addr,lngth,dreg,sreg

where

addr is the address in a target station to write data into

lngth is the number of words (16-bit values) to write, starting at **addr**

dreg is the starting register number to send data from

sreg is the Communication Status Register

The command **PUT** writes a 16-bit word(s) from a specified register to a data table in the PLC. Refer to Section 17.5 for the Communication Status Register information.

Example:

PUT 330,110,#30,#20

This command will write 110 words to address 330 of the target device. The data sent will be copied from registers 30 through 139. Register 20 will contain the Communication Status.

17.6.5 WRITE TO A BIT IN THE TARGET PLC (PUTIO)

Syntax:

PUTIO addr,bit,state,sreg

where

addr is the address in a target station to write data into

bit is the specific bit within addr to modify

state is the bit value to write

0 - RESET

1 - SET

sreg is the Communication Status Register

The command PUTIO sets a specified bit in the target device to the value in an indicated register. The register must be either a "0" or "1". Refer to Section 17.5 for the Communication Status Register information.

Example:

PUTIO 102,10,#30,#20

This command will SET or RESET bit 10 at address 102 of the target device according to the value contained in register 30 (must be 1 or 0). Register 20 will contain the Communication Status.

Chapter 18

4800-E12: GENERAL ELECTRIC PLC NETWORK*

18.1 INTRODUCTION

This chapter contains all of the information that is specific to the 4800-E12: General Electric Series One, Three, Five, and Six PLC interface. The protocol to be used by the module to communicate with the GE PLC is a full-duplex, asynchronous, point-to-point protocol that closely conforms to ANSI X3.28.

The specific commands that allow communications are listed in Section 18.6. There they are discussed in detail, including examples. These commands are:

GETR
GETIS
GETOS
PUTR
PUTOS

The 4800-E12 will communicate directly with Series One, Three, Five, and Six PLCs. The communication standard with the Series One and Three is RS-232, and can be made from the Expansion Card port. The Series Five and Six have the capability of using both the RS-232 and RS-422 communication standards. When using the RS-232 standard, the port on the Expansion Card is used.

When communicating in the RS-422 standard, a Xycom RS-422 Communication Adapter Module must be used. In this case, the port on the Communication Adapter is used for PLC interface, while the port on the Expansion Card is used for peripheral serial communication. The port on the original controller card is not used.

* The 4800-E12 will also directly interface with the Texas Instruments Series 305 PLC.

18.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC through a cable. This is not a standard cable. The connectors must be fabricated. Figures 18-1 through 18-6 illustrate the connections that must be made between the Expansion Card/Communication Adapter port and the PLC.

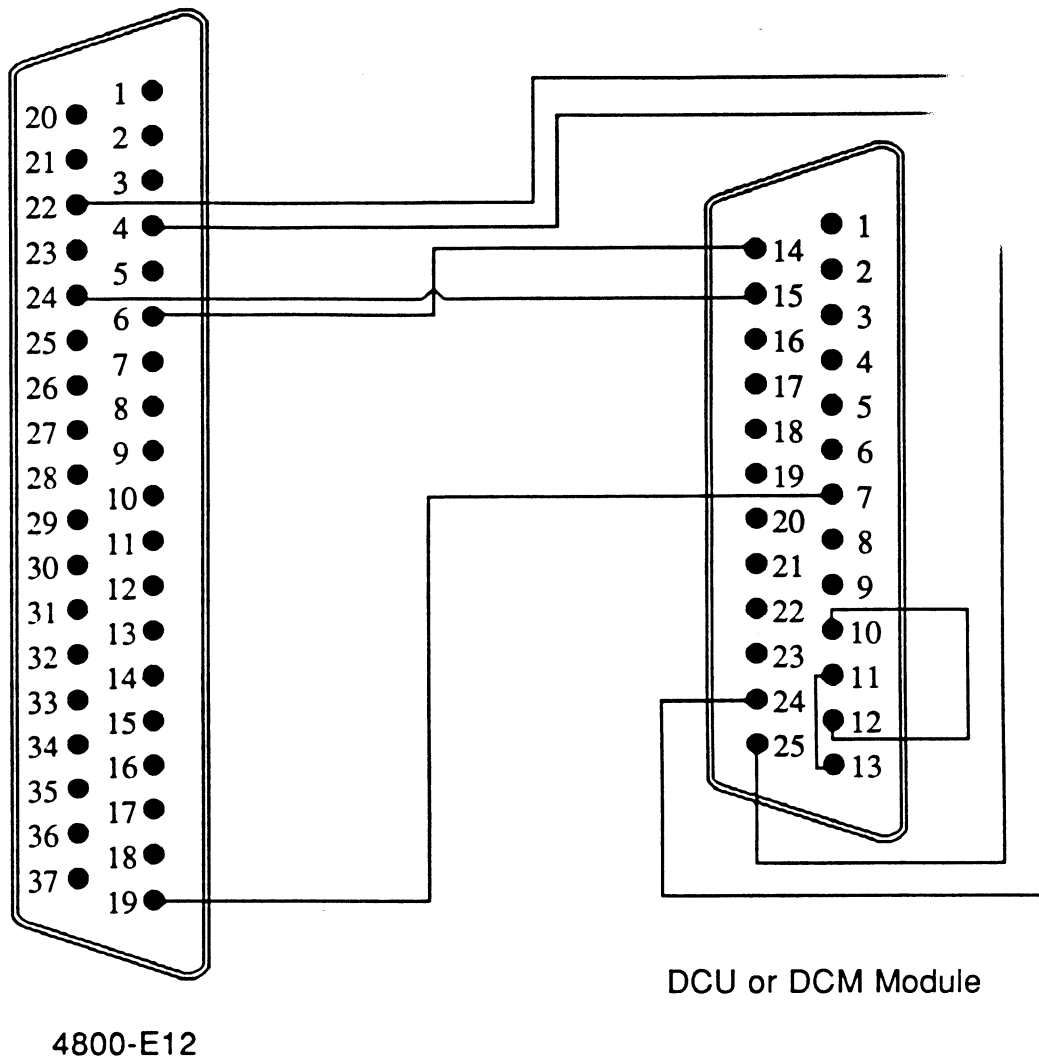


Figure 18-1 GE Series One and Three PLCs

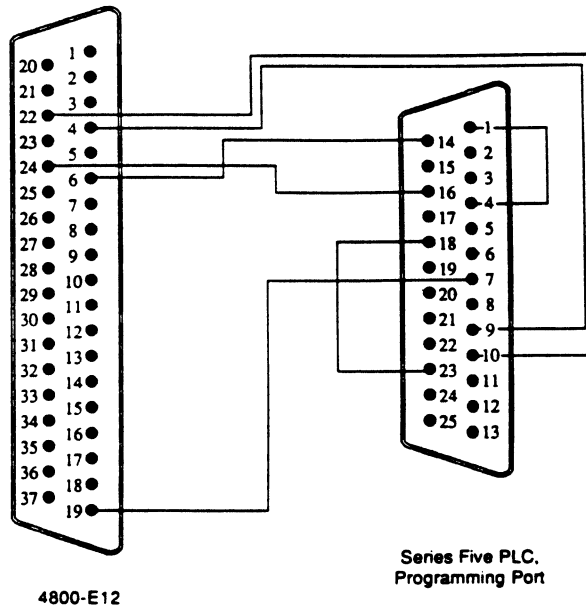


Figure 18-2 GE Series Five PLC
RS-232 Programming Port

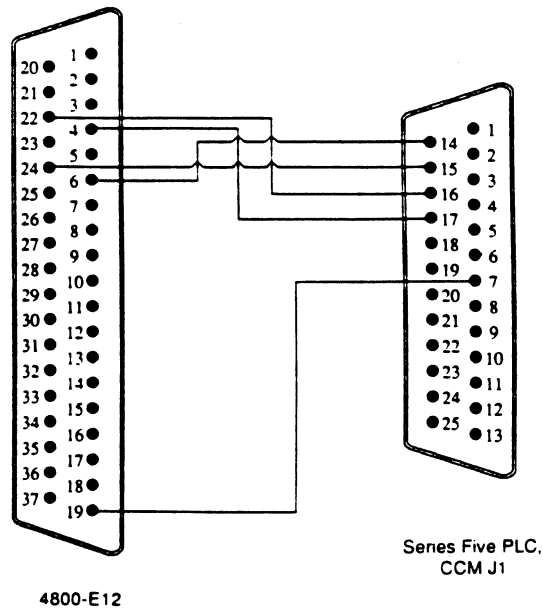


Figure 18-3 GE Series Five PLC
RS-422 CCM J1

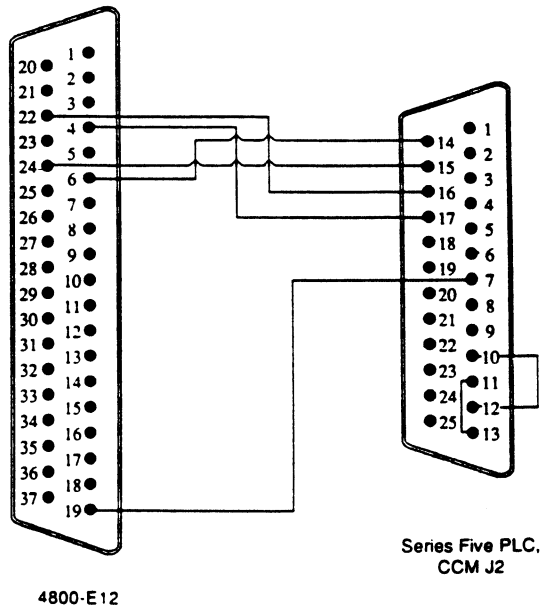


Figure 18-4 GE Series Six PLC
 RS-422 CCM Port J2

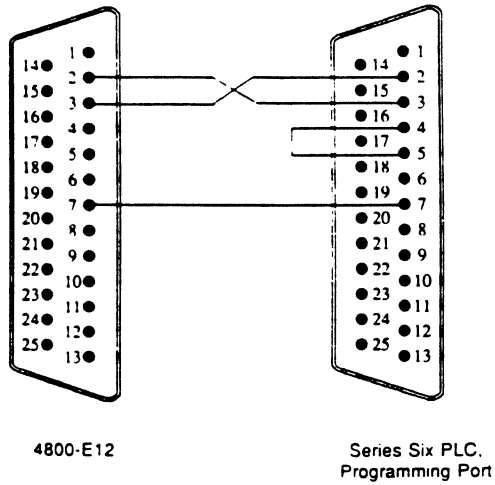
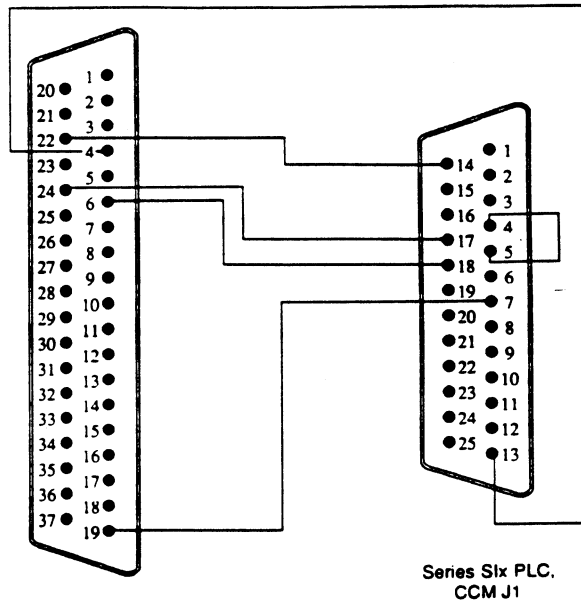
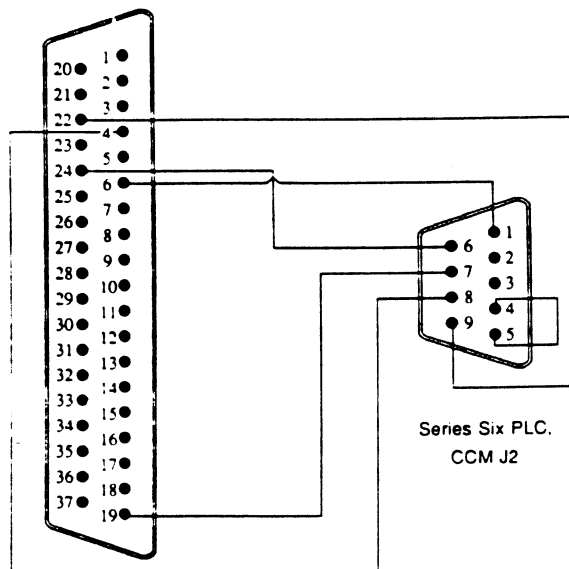


Figure 18-5 GE Series Six PLC
 RS-232 Programming Port



4800-E12

Figure 18-6 GE Series Six PLC
 RS-422 CCM Port J1



4800-E12

Figure 18-7 GE Series Six PLC
 RS-422 CCM Port J2

18.3 CONFIGURATION MENU

The configuration menus are called up from the Main Menu (discussed in Chapter 3). The General Electric Menu looks like the following:

```
-- General Electric Configuration Menu --  
  
6 Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600 7=19.2K  
2 0=No Parity 1=Odd Parity  
50 Source ID# (1-90)  
1 Timeout Value (1-20 seconds)  
  
Use <UP ARROW>, <DOWN ARROW>, <LEFT ARROW>, <RIGHT ARROW>.  
Use values 0 through 9.  
"C" for next configuration menu, <RET> or <ENTER> to quit.
```

Figure 18-8 Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Parity. This option will set the parity used in communication on the Data Highway to either none or even. The type selected should match the other communication device(s).

Source ID#. If the terminal is set up in a network configuration, one or more communication module/PLCs will be used. The Source ID number setting in the General Electric Configuration menu **MUST** be identical to the ID number of the GE PLC CPU to which the terminal is directly connected.

Refer to the General Electric user reference material for information on obtaining the CPU ID number.

Timeout Value. The timeout configuration option determines how long the terminal will wait for a response before it signals a timeout. The value entered will be in seconds, with a range of 1 to 20.

18.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plugs (including the optional RS-422 port).

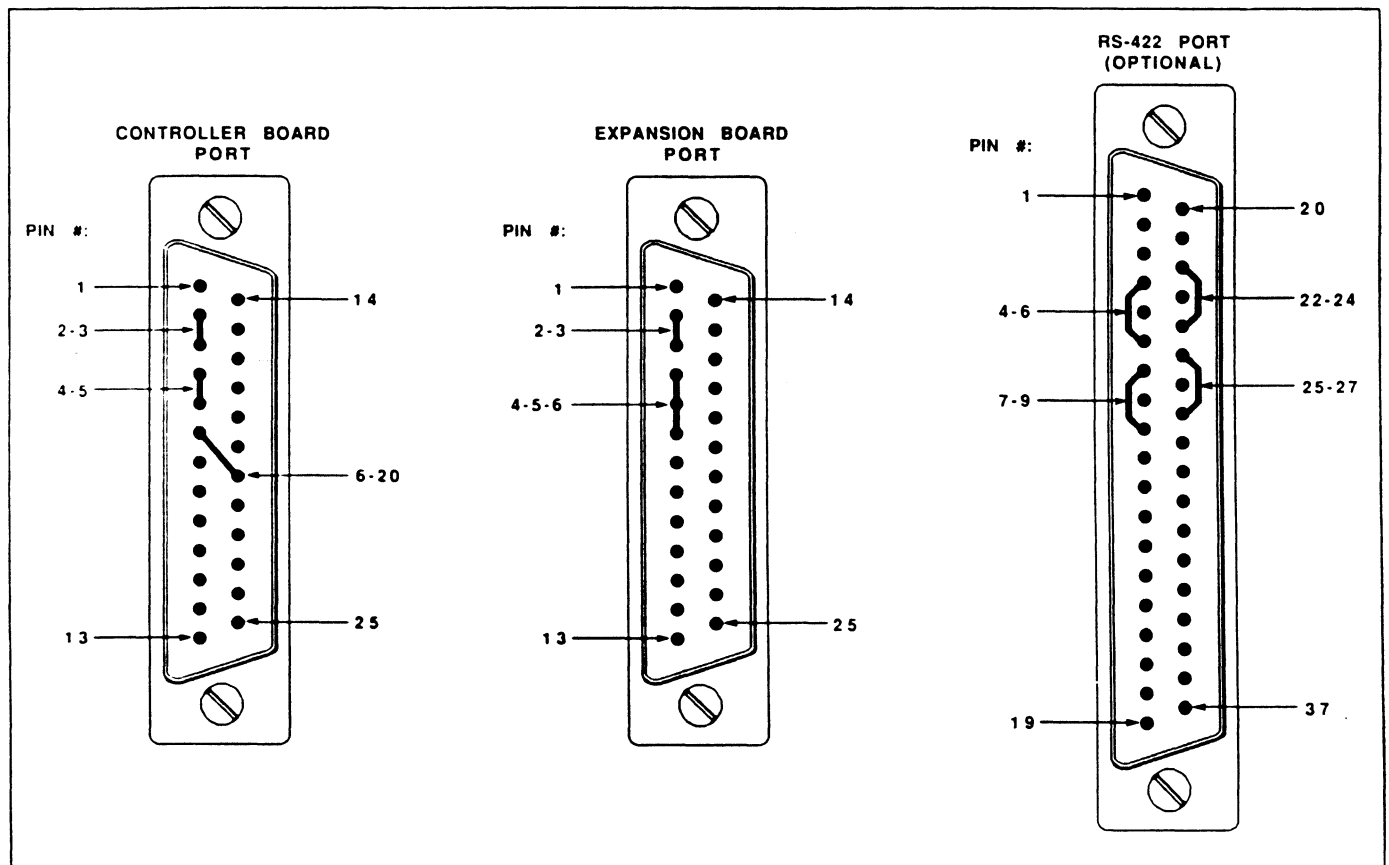


Figure 18-9 Serial Port Test Plugs

18.5 COMMUNICATION STATUS REGISTER

The Communication Status Register contains the status about the message that was just transferred. The Most Significant Byte (MSB) is not used. The Least Significant Byte (LSB) contains information about the message transfer status. A zero value in the entire register indicates a successful transfer.

If the bit noted is set, the condition indicated is true.

Status Register:

MSB: NOT USED

LSB: Message Transfer Status

Bit 0 = Transmit Error
Bit 1 = Receive Error
Bit 2 = Timeout
Bit 3 = Parity Error
Bit 4 = Enquiry Error
Bits 5-7 = NOT USED

18.6 COMMANDS

This section describes the commands that are specific to the 4800-E12: General Electric Series One, Three, Five, and Six PLC Interface.

18.6.1 GET REGISTER (GETR)

Syntax:

```
GETR dest,reg,lngth,dreg,sreg
```

where

dest is the CPU ID number of destination PLC

reg is the register in the target PLC from which the data is to be read

lngth is the number of words to read, starting at addr

dreg is the destination register number to store data

sreg is the Communication Status Register

The command GETR reads data from the Register Table in the PLC. Refer to Section 18.5 for the Communication Status Register information.

Example:

```
GETR 10,20,10,#30,#20
```

This command will read 10 registers starting at register 20 of the target device with ID of 10. The data returned will be stuffed into registers 30 through 39. Register 20 will contain the Communication Status.

18.6.2 GET INPUT STATUS (GETIS)

Syntax:

GETIS dest,addr,lngth,dreg,sreg

where

dest is the CPU ID number of destination PLC

addr is the starting address of Input Status to read

lngth is the number of words to read, starting at addr

dreg is the destination register number to store data

sreg is the Communication Status Register

The command GETIS reads data from the Input Table in the PLC. Refer to Section 18.5 for the Communication Status Register information.

Example:

GETIS 12,40,5,#30,#20

This command will read 5 words of Input Status starting at address 40 of the target device with ID of 12. The data returned will be stuffed into registers 30 through 34. Register 20 will contain the Communication Status.

18.6.3 GET OUTPUT STATUS (GETOS)

Syntax:

GETOS dest,addr,lngth,dreg,sreg

where

dest is the CPU ID number of destination PLC

addr is the starting address of Output Status to read

lngth is the number of words to read, starting at addr

dreg is the destination register number to store data

sreg is the Communication Status Register

The command GETOS reads data from the Output Table in the PLC. Refer to Section 18.5 for the Communication Status Register information.

Example:

GETOS 12,40,5,#30,#20

This command will read 5 words of Output Status starting at address 40 of the target device with ID of 12. The data returned will be stuffed into registers 30 through 34. Register 20 will contain the Communication Status.

18.6.1 PUT REGISTER (PUTR)

Syntax:

PUTR dest,reg,lngth,creg,sreg

where

dest is the CPU ID number of destination PLC

reg is the register in the target PLC from which the data is to be written

lngth is the number of words to write, starting at addr

dreg is the destination register number to send data from

sreg is the Communication Status Register

The command PUTR reads data to a register in the Register Table in the PLC. Refer to Section 18.5 for the Communication Status Register information.

Example:

PUTR 15,33,13,#30,#20

This command will write 13 words of data starting at register 33 of the target device with ID of 15. The data returned will be stuffed into registers 30 through 42. Register 20 will contain the Communication Status.

18.6.3 PUT OUTPUT STATUS (PUTOS)

Syntax:

PUTOS dest,addr,lngth,dreg,sreg

where

dest is the CPU ID number of destination PLC

addr is the starting address of Output Status to write

lngth is the number of words to write, starting at addr

dreg is the destination register number to write data from

sreg is the Communication Status Register

The command PUTOS writes a word of output to the Output Table in the PLC. Refer to Section 18.5 for the Communication Status Register information.

Example:

PUTOS 12,40,1,#30,#20

This command will write 1 word of Output Status starting at address 40 of the target device with ID of 12. The data returned will be copied from register 30. Register 20 will contain the Communication Status.

Chapter 19

4800-E13: SIEMENS PLC NETWORK

19.1 INTRODUCTION

This chapter contains all of the information that is specific to the 4800-E13: Siemens S5-1150 PLC interface.

The specific commands that allow communications are listed in Section 19.6. There they are discussed in detail, including examples. These commands are:

GET
GETDB
GETTC
PUTDB
COMDB

19.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC through a cable. This is not a standard cable. The connectors must be fabricated. Figure 19-1 below illustrates the connections that must be made between the Expansion Card port and the PLC.

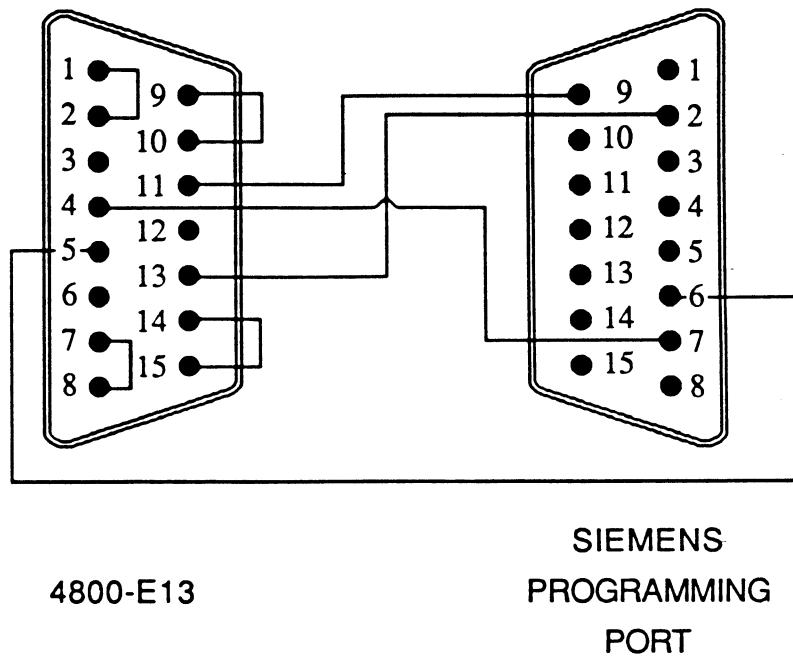


Figure 19-1 Connections

19.3 CONFIGURATION MENU

On other Xycom 4800-series Expansion Modules, there is a third Direct Connect Port Configuration Menu. This configuration is fixed on the 4800-E13, and cannot be altered by the user. The values are as follows:

Baud = 9600
Stop Bits = 1
Data Bits = 8
Parity = Even

19.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plugs.

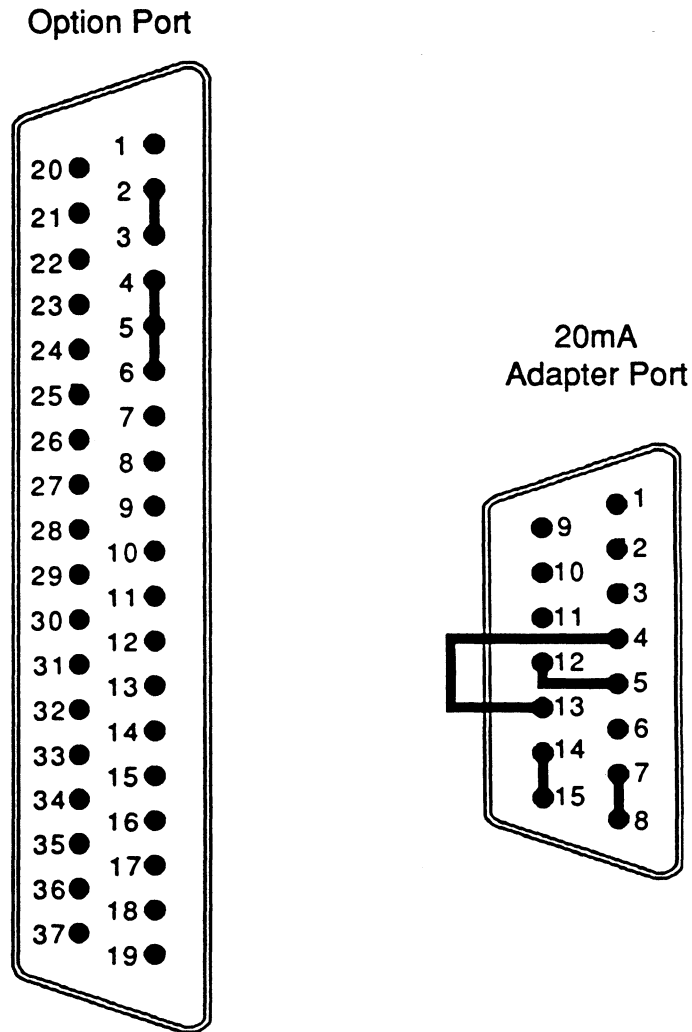


Figure 19-2 Serial Port Test Plugs

19.5 COMMUNICATION STATUS REGISTER

The Communication Status Register in the OIL command contains the status of the message previously transferred. If the OIL register specified is set then the condition corresponding to that value described below is true.

Status Register:

0	No errors
1	Unable to establish communications
2	Timeout
3	Parity error
4	Response error
5	Data block does not exist
6	Write error

19.6 COMMANDS

This section describes the commands that are specific to the 4800-E13 Siemens PLC interface.

The following lists the address types within the Siemens PLC that can be read by the 4800-E13 Expansion Module.

<u>TYPE</u>	<u>RANGE(b)</u>	<u>DESCRIPTION</u>
Ib.x	0 to 127	input image (PII)
Ob.x	0 to 127	output image (PIQ)
Fb.x	0 to 255	flags (F)

where

b is the byte number in that address type (valid range given in column 2)
x is the bit number within the specified byte.

19.6.1 GET PLC DATA (GET)

Syntax:

GET type,addr,lngth,dreg,sreg

where:

type is the number identifying what type of memory to read:
0 = input image (PII)
1 = output image (PIQ)
2 = flags (F)

addr is the Address of the block to read (see below).

lngth is the Length (in 16-bit words) of the read.

dreg is the Initial Destination Register for value(s).

sreg is the Communication Status Register.

The command GET will read data from the PLC and store it in specified OIL registers. The GET command will read only byte values from the PLC. The user will have to use the various OIL bit commands to check a specific bit.

Example:

GET 1,0,128,#100,#229

This command will read 128 output image values beginning at byte 0. The data read will be stuffed into registers 100 through 228. Register 229 will contain the Communication Status.

19.6.2 GET DATA BLOCK (GETDB)

Syntax:

GETDB block,maxlngth,dreg,lreg,sreg

where:

block is the number of the data block to read (2 to 255).

maxlngth is the maximum number of words that will be stored into the OIL registers.

dreg is the Initial Destination Register for value(s).

lreg is the Length Register containing the actual size of the data block.

sreg is the Communication Status Register.

The command GETDB will read a data block from the PLC and store it beginning with a specified OIL register. The length of the data block refers only to the length of the data area and does not include the header. The header is not stored.

Example:

GETDB 2,1,#15,#16,#17

This command will read the first word from data block 2. The data read will be stuffed into register 15. Register 16 will store the block length read. Register 17 will contain the Communication Status.

19.6.3 GET TIMER/COUNTER DATA (GETTC)

Syntax:

GETTC type,addr,lngth,TCsreg,TCvreg,sreg

where:

type is the data read:

0 = timer

1 = counter

addr is the initial timer/counter to read.

lngth is the number of timers/counters to read.

TCsreg is the Initial Destination for the timer/counter Status data.

TCvreg is the Initial Destination for the timer/counter Value data.

sreg is the Communication Status Register.

TCsreg and TCvreg will store two sections of the timer/counter data word. Bits 15 - 10 contain the timer/counter status data. Bits 9 - 0 contain the current timer/counter value.

The command GETTC will read a either the timer or counter status and value data from the PLC and store them in specified OIL registers.

Example:

GETTC 0,0,1,#20,#30,#40

This command will read the data from timer 0. The timer status will be stuffed into register 20, while register 30 will hold the timer value. Register 40 will contain the Communication Status.

19.6.4 PUT DATA BLOCK (PUTDB)

Syntax:

PUTDB block,lngth,source,sreg

where:

block is the number of the data block to write (2 to 255).

lngth is the maximum number of words that will be stored into the PLC.

source is the initial register to copy that data for the data block.

sreg is the Communication Status Register.

The command PUTDB will read data from the specified OIL register(s) to create a data block to write to the PLC. The data block refers only to the length of the data area and does not include the header. A header is automatically built when the block is created.

Example:

PUTDB 2,1,#13,#14

This command will read one word from OIL register 13 and create data block 2. Register 14 will contain the Communication Status.

19.6.5 COMPRESS DATA BLOCK (COMDB)

Syntax:

COMDB sreg

where:

sreg is the Communication Status Register.

The command COMDB will compress the data block memory area in the PLC. This command should be issued when the PUTDB command returns with a write error.

Example:

COMDB #13

This example will compress the data block area. Register 13 will contain the Communication Status.

Chapter 20

4800-E14: ALLEN-BRADLEY T3 PORT

20.1 INTRODUCTION

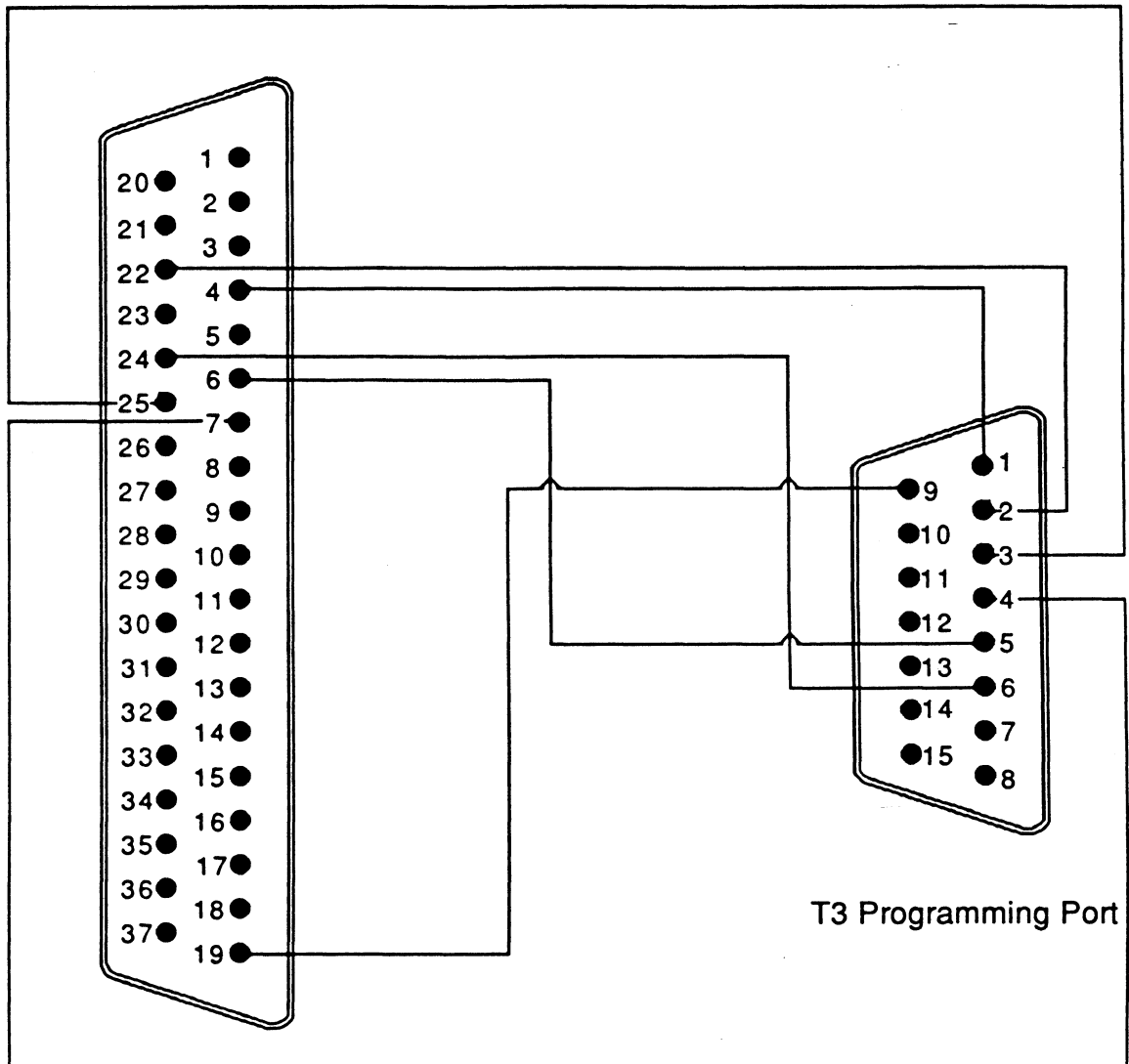
This chapter contains all of the information that is specific to the 4800-E14: Allen-Bradley T3 Programming Port interface. The protocol used by the module to communicate with the A-B PLC is a full-duplex, asynchronous, point-to-point protocol that closely conforms to ANSI X3.28.

The specific commands that allow communications are listed in Section 20.6. There they are discussed in detail, including examples. These commands are:

GET
GETIO
GETDIAG
PUT
PUTIO

20.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC through a cable. This is not a standard cable. The connectors must be fabricated. Figure 20-1 below illustrates the connections that must be made between the Expansion Card port and the PLC.



4800-E14

Figure 20-1 Connections

20.3 CONFIGURATION MENU

On other Xycom 4800-series Expansion Modules, there is a third Direct Connect Port Configuration Menu. This configuration is fixed on the 4800-E14, and cannot be altered by the user. The values are as follows:

Baud = 9600
Parity = Even

20.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plugs.

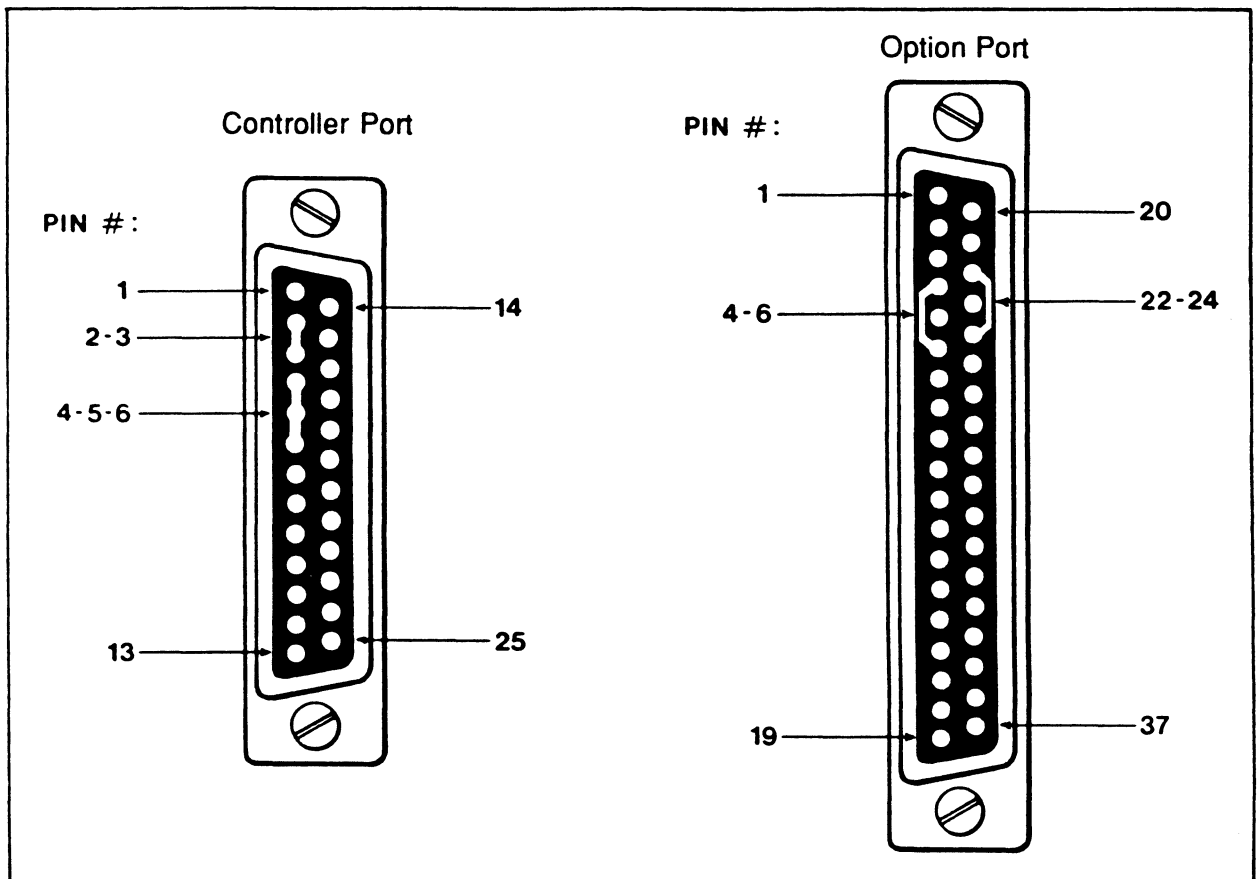


Figure 20-2 Serial Port Test Plugs

20.5 COMMUNICATION STATUS REGISTER

The Communication Status Register in the OIL command contains the status of the message previously transferred. The data value returned in the message from the Communication Module will denote a positive decimal digit. If a value is returned in the Status Register then the condition corresponding to that value described below is true.

Status Register:

0	No errors
1	Response error
2	Timeout
3	Parity error

20.6 COMMANDS

This section describes the commands that are specific to the 4800-E6 Allen-Bradley Data Highway Interface.

20.6.1 GET BLOCK (GET)

Syntax:

GET addr,lngth,dreg,sreg

where:

addr is the Octal Address of the block to read

lngth is the Length of the read (≥ 1)

dreg is the Initial Destination Register for value(s)

sreg is the Communication Status Register

The command GET will read a block of one or more words from the data table in an Allen-Bradley PLC and store it in a specified OIL register. Refer to Section 20.6 for the Communication Status Register information.

Example:

GET 101,3,#21,#20

This command will read the block of information in data table locations 101 through 103 of the PLC. The data read will be stuffed into registers 21 through 23. Register 20 will contain the Communication Status.

20.6.2 GET BIT (GETIO)

Syntax:

GETIO addr,lbit,dreg,sreg

where:

addr is the Octal Address of the block to read

lbit is the bit number (0-15)

dreg is the Initial Destination Register for value(s)

sreg is the Communication Status Register

The command GETIO will read a specified bit in a word from the input table in an Allen-Bradely PLC. Refer to Section 20.6 for the Communication Status Register information.

Example:

GETIO 1,2,#21,#20

This command will read bit 2 of the word at address 1 of the PLC. The data read will be stuffed into register 21. Register 20 will contain the Communication Status.

20.6.3 GET DIAGNOSTIC STATUS (GETDIAG)

Syntax:

GETDIAG dreg,sreg

where:

dreg is the Initial Destination Register for value

sreg is the Communication Status Register

The command GETDIAG will read the diagnostic status (size of the data table) from the Allen-Bradley PLC and return this to a specified register. Refer to Section 20.6 for the Communication Status Register information.

Example:

GETDIAG #21,#20

This command will return the diagnostic status to register 21. Register 20 will contain the Communication Status.

20.6.4 PUT BLOCK (PUT)

Syntax:

puT addr,lngth,dreg,sreg

where:

addr is the Octal Address of the block to write

lngth is the Length of the write (≥ 1)

dreg is the Initial Source Register for value(s)

sreg is the Communication Status Register

The command PUT will write a block of one or more words from a specified OIL register to a location in the data table in an Allen-Bradley PLC. Refer to Section 20.6 for the Communication Status Register information.

Example:

PUT 101,3,#21,#20

This command will write the block of information contained in OIL register 21 through 23 into data table locations 101 through 103 of the PLC. Register 20 will contain the Communication Status.

20.6.5 PUT BIT (PUTIO)

Syntax:

PUTIO addr,lbit,dreg,sreg

where:

addr is the Octal Address of the block to write

lbit is the bit number (0-15)

dreg is the Initial Source Register for value(s)

sreg is the Communication Status Register

The command PUTIO will write read a specified bit value (0/1) from an OIL register to a word in the data table in an Allen-Bradely PLC. Refer to Section 20.6 for the Communication Status Register information.

Example:

PUTIO 1,2,#21,#20

This command will write the logical value (0/1) in register 21 to bit 2 of the word at address 1 of the target device. Register 20 will contain the Communication Status.

Chapter 21

4800-E15: CINCINNATI MILICRON

22.1 INTRODUCTION

No information is available at this time for the 4800-E15: Cincinnati Milicron PLC interface.

Chapter 22

4800-E16: TEXAS INSTRUMENTS SERIES 405 PLC

22.1 INTRODUCTION

This chapter contains all of the information that is specific to the 4800-E16: Texas Instruments Series 405 PLC interface. The protocol to be used by the module to communicate with the Texas Instruments Series 405 PLC is the K-sequence protocol.

The specific commands that allow communications are listed in Section 22.6. There they are discussed in detail, including examples. These commands are:

GET
PUT

22.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC through a cable. This is not a standard cable. The connectors must be fabricated. Figure 21-1 below illustrates the connections that must be made between the Expansion Card port and the PLC.

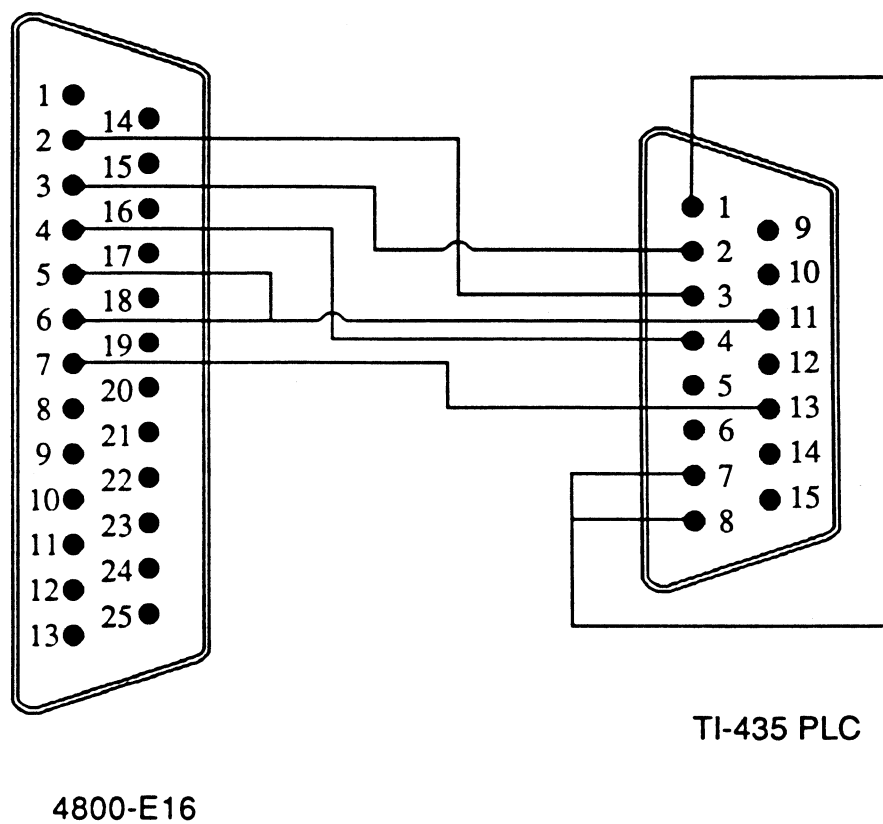


Figure 21-1 Connections

22.3 CONFIGURATION MENU

On other Xycom 4800-series Expansion Modules, there is a third Direct Connect Port Configuration Menu. This configuration is fixed on the 4800-E16, and cannot be altered by the user. The values are as follows:

Baud = 9600
Stop Bits = 1
Data Bits = 8
Start Bits = 1
Parity = Even

22.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plugs.

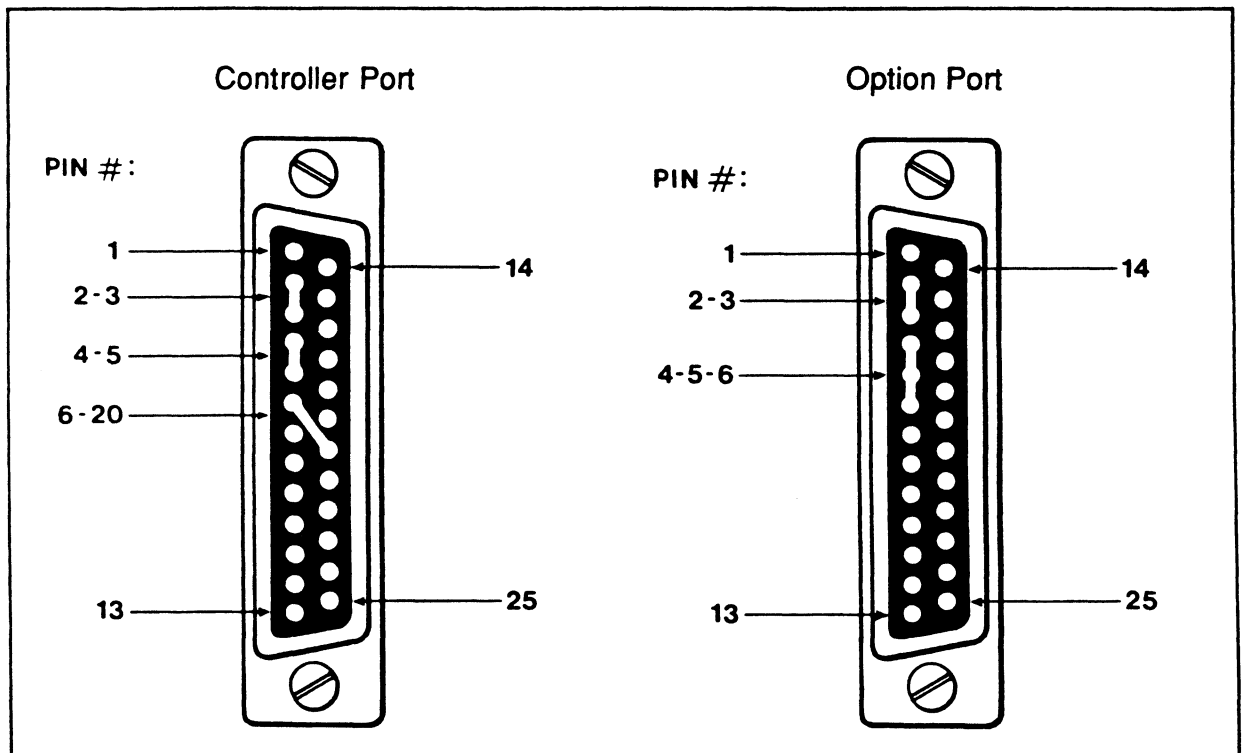


Figure 22-2 Serial Port Test Plugs

22.5 COMMUNICATION STATUS REGISTER

The Communication Status Register contains the status about the message that was just transferred. The Most Significant Byte (MSB) is the status returned from the Texas Instruments PLC. The Least Significant Byte (LSB) contains information about the message transfer status. A zero value in the entire register indicates a successful transfer.

If the bit noted is set, the condition indicated is true.

Status Register:

MSB: PLC Response

C0H	successful GET for memtypes 0-11
C6H	successful PUT for memtypes 0-11
D0H	successful GET for memtype 12
D1H	successful PUT for memtype 12
FFH	unsuccessful transfer

LSB: Message Transfer Status

Bit 0	Transmit Error
Bit 1	Receive Error
Bit 2	Timeout
Bit 3	Parity Error
Bit 4	Overrun error
Bit 5	Framing Error
Bit 6	Enquiry error
Bit 7	NOT USED

22.6 COMMANDS

This section describes the commands that are specific to the 4800-E6 Allen-Bradley Data Highway Interface.

Both the GET and PUT commands use a variable called memtype. This refers to the memory type within the Texas Instruments Series 405 PLC that can be accessed by the Xycom terminal. These types, symbols, lengths, and descriptions are given in the table below. Please refer to the TI-405 technical manual for exact memory mapping details.

Table 22-1 Memory Type

TYPE	SYMBOL	LENGTH	DESCRIPTION
0	T	128	Timer Accumulators
1	CT	128	Counter Accumulators
2	U	3228	User Registers
3	GX	32	Remote I/O Memory
4	X	20	Input Memory
5	Y	20	Output Memory
6	C	30	Control Relay
7	S	24	Stage Memory
8	TI	8	Timer Relay
9	CT1	8	Counter Relay
10	SP1	5	Special Relays 1
11	SP2	10	Special Relays 2
12	V	17056	Anywhere in V Memory
13	SPAD		Anywhere in Scratch Pad Memory

22.6.1 GET DATA (GET)

Syntax:

GET memtype,addr,lngth,dreg,sreg

where

memtype is the number identifying type of memory to read (see Section 22.6)

addr is the address of the block to read

lngth is the number of words to read, starting at addr

dreg is the initial destination register number to store data

sreg is the Communication Status Register

The command GET will read 16-bit word(s) from a specified memory location in the TI-405 and place the result in a specified OIL register. Refer to Section 22.5 for the Communication Status Register information.

Example:

GET 2,10,20,#50,#70

This command will read 20 words of data starting at address 10 of the "User Register" memory area of the TI-405 PLC. The data returned will be stuffed into registers 50 through 69. Register 70 will contain the Communication Status.

22.6.2 PUT DATA (PUT)

Syntax:

PUT memtype,addr,lngth,dreg,sreg

where

memtype is the number identifying type of memory to write (see Section 22.6)

addr is the address of the block to read

lngth is the number of words to read, starting at addr

dreg is the initial source register number to copy

sreg is the Communication Status Register

The command PUT will write 16-bit word(s) from a specified OIL register to a memory location in the TI-405. Refer to Section 22.5 for the Communication Status Register information.

Example:

PUT 2,10,20,#70,#90

This command will write the data contained in OIL registers 70 through 90 to locations 10 through 19 of the "User Register" memory area of the TI-405 PLC. Register 90 will contain the Communication Status.

Chapter 23

4800-E17 OMRON PLC Expansion Module

23.1 INTRODUCTION

This appendix contains information specific to the 4800-E17 Omron PLC interface to a Xycom 4800 series terminal.

The specific commands that allow communications are described in Section 23.5 and listed below for reference:

PUT
GET
GETER
INIT

The 4800 supports both Single-Link and Multi-Link communications configurations. The Single-Link configuration is for a single PLC connected to a 4800 Terminal. The Multi-Link configuration allows up to 31 PLCs to be connected to a 4800 Terminal. You select the appropriate configuration in the Omron Port Configuration Menu.

The direct connect supports the OMRON single-frame format. The PLC interface limits the maximum size of a single-frame message to 131 characters. This limits the operator to a maximum of 29 channels that can be read or written at one time.

23.2 CABLING THE TERMINAL TO THE PLC

The Omron PLC connects to the controller of the 4800 via a non-standard cable. The 4800 can work on either RS-232 or RS-422 cable, depending on which port is installed. You must construct the connectors, according to the pinouts shown in Figures 23-1 and 23-2.

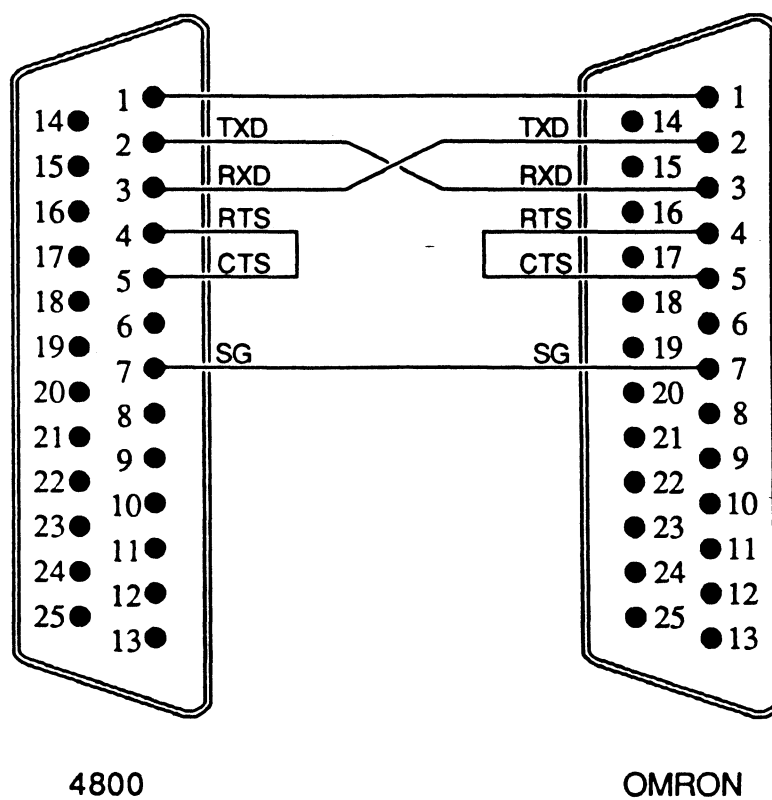


Figure 23-1. RS-232 Connection to Omron PLC

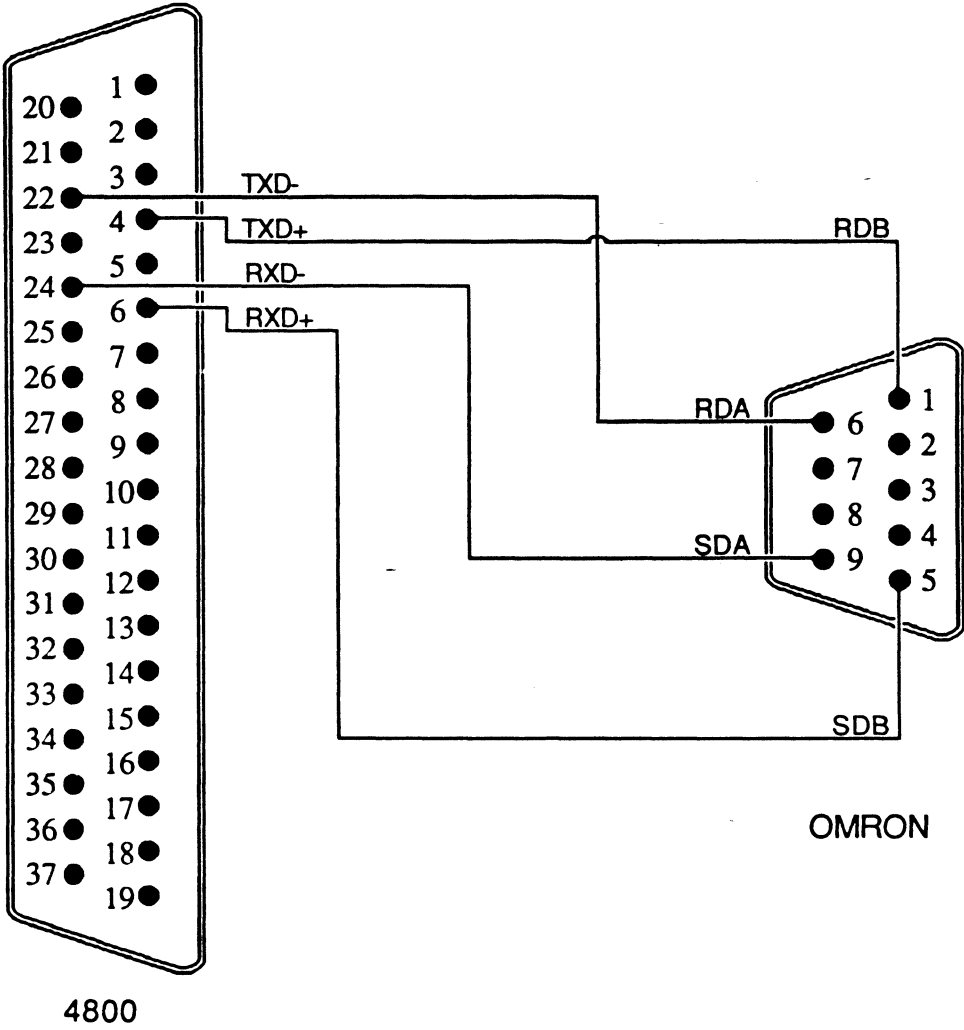


Figure 23-2. RS-422 Connection to Omron PLC

23.3 CONFIGURATION MENU

The configuration menu is called up from the Main Menu (discussed in Chapter 3). The OMRON menu is shown in Figure 23-4.

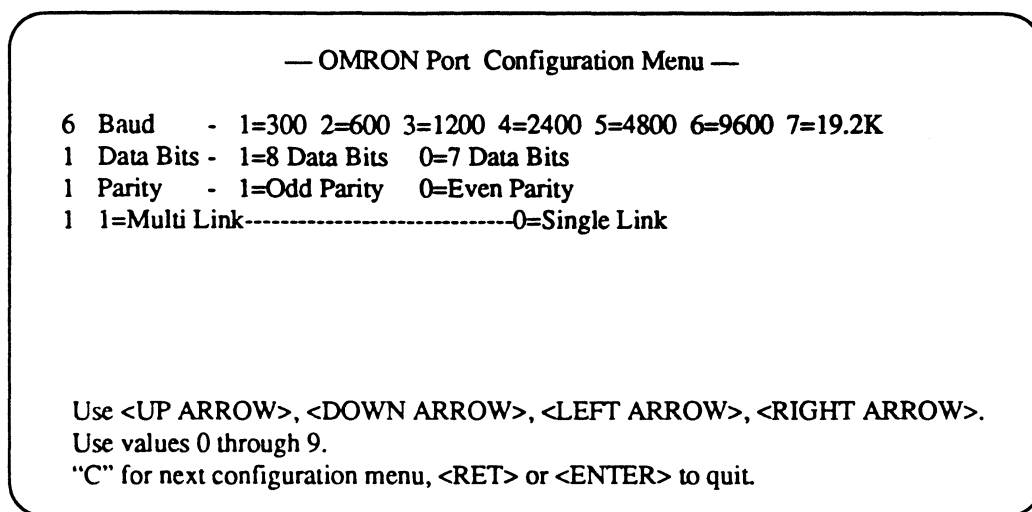


Figure 23-3. Omron Port Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Data Bits. The data bit option should be set to match that of the PLC.

Parity. This option sets the parity used in communication to the PLC to odd or even; it should be set to match the parity of the PLC.

Multi/Single Link. The Single Link configuration signifies that only one PLC is connected to your 2000 Terminal; the Multi Link configuration allows for up to 31 PLCs connected to the 2000.

23.4 COMMUNICATION STATUS REGISTER

The status value is placed in the OIL register as specified by the command line parameter Sreg. The most significant byte (MSB) contains the response code returned from the OMRON HLU. The least significant byte (LSB) contains Xycom internal error bits. The first four bits are unused and set to zero.

Table 23-1 below describes the condition corresponding to each bit set in the LSB of the status register:

Table 23-1. Xycom Terminal Status Codes

LSB Bit #	Status From Xycom Terminal
0-3	Not Used
4	Transmit Error
5	Receive Error
6	Frame Check Sequence
7	Timeout

Table 23-2 shows the response codes returned by the OMRON HLU.

Table 23-2. OMRON Status Register Response Codes

Hex Value of MSB	Omron Status Response
00	Normal completion
01	Not executable in RUN mode
02	Not executable in MONITOR mode
03	Not executable with PROM mounted
04	Address over (data overflow)
0B	Not executable in PROGRAM mode
0C	Not executable in PROM mounted
0D	Not executable in LOCAL mode
10	Parity error
11	Framing error
12	Overrun
13	FCS error
14	Format error (parameter length error)
15	Entry number data error (parameter error, data code error, data length error)
16	Instruction not found
18	Frame length error
19	Not executable (unexecutable error clear, non-registration of I/O table, etc.)
20	I/O table generation impossible (unrecognized Remote I/O Unit, channel over, duplication of Optical Transmitting I/O Unit)
A0	Aborted because of parity error in transmit data
A1	Aborted because of framing error in transmit data
A2	Aborted because of overrun in transmit data
A4	Aborted because of format error in transmit data
A5	Aborted because of entry number data error in transmit data
A8	Aborted because of frame length error in transmit data
B0	Not executable because program area is not 16 Kbytes

23.5 COMMANDS

This section describes the OIL commands specific to the 4800-E17 Omron interface.

PUT:	Write to PLC register areas
GET:	Read from PLC register areas
GETER:	Read and Clear HLU error conditions
INIT:	Abort current operations and re-initialize the HLU

Register areas include:

IR:	Input/Output Relay area channels
HR:	Holding Relay area channels
AR:	Auxiliary Relay area channels
DM:	Data Memory area channels
TC:	Timer/Counter area channels

NOTE

The direct connect does internal range checking, and will not allow writes to output relays.

23.5.1 Read and Clear Host Link Unit Error Conditions (GETER)

Syntax:

GETER erclr,dreg,sreg,[unitno]

where

erclr = 0 for Error is not cleared, 1 for Error is cleared.
dreg = starting register in which data will be stored.
sreg = communication status register.
unitno = PLC's Unit Number. This value is only used if a multi-link system is
 in use.

Example:

GETER 0,#90,#70

This example checks to see if an error has occurred without clearing it. The result will be stored in two words starting at OIL register #90. Register #70 will contain the communication status. A single-link system is used.

Table 23-3. Error Response Codes

First Word

Bit #	Error
15	FALS (CPU stops)
14	End Instruction Missing (FO)/Program Error (F3)
13	Host Link Unit transmission error
12	RTI Instruction Error
11	PC Link Error
10	I/O Bus Error (CO to 4)
9	JMP Instruction Error (F2)
8	Memory Error (F1)
7	FAL Error
6	Battery Failure (F7)
5	Duplex Bus Error
<u>4</u> <u>3</u> <u>2</u>	
0 0 0	CPU Rack
0 0 1	Expansion Rack 1
0 1 0	Expansion Rack 2
0 1 1	Expansion Rack 3
1 0 0	Expansion Rack 4
1 0 1	Expansion Rack 5
1 1 0	Expansion Rack 6
1 1 1	Expansion Rack 7
<u>1</u> <u>0</u>	Data from I/O Bus
0 0	Group 1 (control signal error)
0 1	Group 2 (data bus failure)
1 1	Group 3 (address bus failure)

Table 23-3. Error Response Codes (continued)

Second Word

Bit #	Error
15	Remote I/O Error (B0 to 3) I/O setting error (E0) Indirect JMP Instruction Error (F9) Scan time over (F8/I/O Unit over (E1) I/O verify error (E7) DM CH error (F8)
14	
13	
12	
11	
10	
9	
8	
7	FAL, FALS number* 0 .. \$0 9 .. \$9
6	
5	
4	
3	FAL, FALS number* 0 .. \$0 9 .. \$9
2	
1	
0	

*Refer to Omron documentation for FAL, FALS error codes.

23.5.2 Abort Current Operations and Re-Initialize Host Link Unit (INIT)

Syntax:

INIT

Example:

INIT

This example will send a message to abort the process being performed by the HLU and enable reception of the next command. It will then initialize the transmission control procedure of all the PLCs connected to the host computer.

23.5.3 Read from PLC Register Areas (GET)

Syntax:

GET area,bch,len,dreg,sreg,[unitno]

where

- | | |
|--------|--|
| area | = indicates which area to read. Options are 1, 2, 3, 4, 5 (which stand for IR, HR, AR, DM, and TC respectively). |
| bch | = beginning OMRON channel within area to start reading from. The driver will ensure that valid ranges for each area are entered. The valid ranges are listed below:

IR - 000-246.
HR - 00-99.
AR - 00-22.
DM - 0000-1999.
TC - 000-511. |
| len | = the number of OMRON channels to read. |
| dreg | = starting OIL register number into which received data will be stored. |
| sreg | = communication status register |
| unitno | = PLC's Unit Number. This value is only used if a multi-link system is in use. |

Example:

READ 4,40,2,#50,#90,2

This command will read two channels of data starting from channel 40 of the DM (Data Memory) area. The data will be stored in registers #50 and #51. Register #90 will contain the communication status. A multi-link system is used.

23.5.4 Write to PLC Register Areas (PUT)

Syntax:

PUT area,bch,len,dreg,sreg,[unitno]

where

- area** = indicates which area to write. Options are 1, 2, 3, 4, 5 (which stand for IR, HR, AR, DM, and TC respectively).
- bch** = beginning OMRON channel to start writing to. The driver ensures that valid ranges for each area are entered. The valid ranges are listed below:
- IR - 30-49 and 232-246.
 - Channels 0-29 and 50-231 are used as outputs.
 - HR - 00-99
 - AR - 07-22
 - DM - 0000-0999
 - TC - 000-511
- len** = number of OMRON data channels to write.
- dreg** = starting OIL register to take data from.
- sreg** = communication status register.
- unitno** = PLC's Unit Number. This value is only used if a multi-link system is in use.

The direct connect does internal range checking and will not allow writes to output relays.

Example:

WRITE 3,1,2,#44,#90,2

This example writes two OIL registers (#44 and #45) to the AR area beginning at channel 1. OIL register #90 will contain the communication status. Multi-link system is used.

4800-E18: MITSUBISHI MELSEC-A

24.1 INTRODUCTION

This chapter contains the information specific to the 4800-E18: Mitsubishi MELSEC-A PLC interface. The protocol this module uses to communicate with the Mitsubishi MELSEC-A is full-duplex, asynchronous, RS-232C.

The specific commands that allow communication are fully described in Section 24.5. They are:

GETDB
GETDW
GETBUF
GETCPU
GETPAR
GETSFU
PUTDB
PUTDW
PUTBUF
PUTCPU
PUTPAR
PUTSFU

This direct connect communicates with the Mitsubishi MELSEC-A PLC using the AJ71C24 unit. Because multiple PLC stations can be networked to Xycom's 4800-E18, the specific station number must be specified in the direct connect commands.

NOTE

AJ71C24 refers to the Mitsubishi Computer Link Unit and PC CPU refers to the Internal Mitsubishi PLC CPU (A1CPU, A2CPU, or A3CPU).

The 4800-E18 direct connect can access data in the following PLC memory areas:

- Device memory (bit and word devices)
- Buffer memory (AJ71C24 - PC CPU)
- PC CPU (type read, run/stop request)
- Parameter memory (PC CPU)
- Special Function Unit (SFU) buffer memory

24.2 CABLING THE TERMINAL TO THE PLC

The direct connect cable between the Mitsubishi MELSEC-A PLC's AJ71C24 computer link unit and the XYCOM 4800-E18 is shown in Figure 24-1 below:

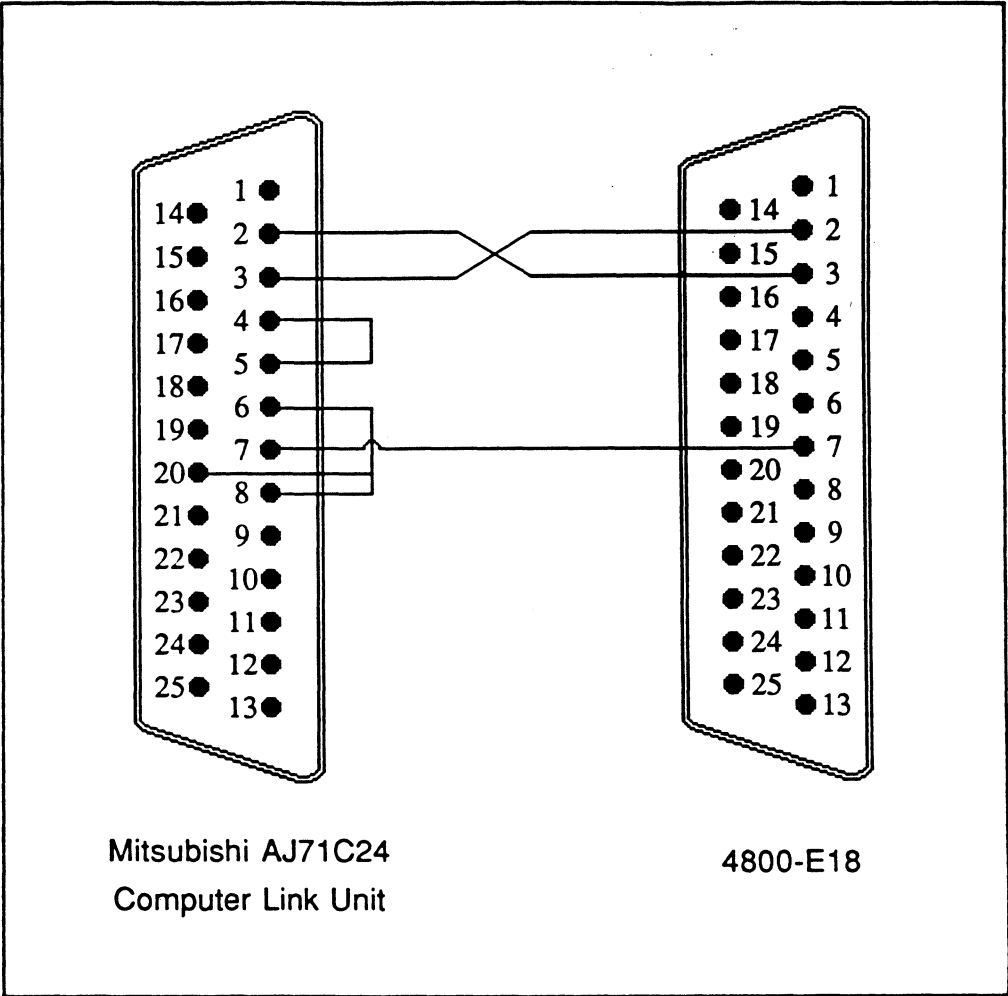


Figure 24-1. Connections

24.3 CONFIGURATION MENU

Most of the communication parameters are switch-selectable on the AJ71C24, and can be selected for the controller board in the Mitsubishi MELSEC-A PLC Configuration Menu. This direct connect will use 1 start bit and 1 stop bit. It supports communication with the controller board port using RS-232C.

The Configuration Menu is called up from the Main Menu (discussed in chapter 3). The Mitsubishi configuration menu is shown in Figure 24-2 below:

```
-- Mitsubishi MELSEC-A PLC Configuration Menu --  
  
6 Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600 7=19.2K  
1 Parity - 0=No Parity 1=Odd Parity 2=Even Parity  
1 1=8 Data Bits ----- 0=7 Data Bits  
1 Format of Control Protocol (1-4)  
0 Wait for message delay (0-9, unit=10 ms)  
  
Use <UP ARROW>, <DOWN ARROW>, <LEFT ARROW>, <RIGHT ARROW>.  
Use values 0 through 9.  
"C" for next configuration menu, <RET> or <ENTER> to quit.
```

Figure 24-2. Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Parity. This option will set the parity used in communication on the Mitsubishi to none, odd, or even. The type selected should match the other communication device(s).

Data Bits. This option will select 7 or 8 data bits per character. This should match the data bits of the other communication device(s).

Control Protocol. This option can be set to the Mitsubishi AJ71C24 specific protocol format (1, 2, 3, or 4). Refer to the Mitsubishi MELSEC-A User Manual for specific information.

Message Delay. This option sets the message delay. Each unit is equal to 10 ms. For example, selecting 1 will set the message delay to 10 ms, while selecting 2 sets the delay to 20 ms.

NOTE
The 4800-E18 direct connect does not support the simultaneous selection of 7 bits per character and no parity.

24.4 COMMUNICATION STATUS REGISTER

The Communication Status Register is a user-specified OIL register which contains the status about the message that was just transferred. The Most Significant Byte (MSB) contains information about error codes from the 4800-E18 direct connect software. The Least Significant Byte (LSB) contains information about AJ71C24 NAK error codes issued.

Since multiple MSB errors may occur simultaneously, the values shown must be or'ed to compare to the status value returned. For example, if a time out error occurs, a receive error will also be flagged and the MSB communication status reported will be 06H.

On a AJ71C24 NAK response, the MSB communication status bit 1 is reported set and the value of the LSB will clarify the reason for the NAK response.

COMMUNICATION STATUS WORD

ERROR BYTE DEFINITION

<u>MSB</u>	<u>LSB</u>
------------	------------

Bit 0
Bit 1
Bit 2
Bit 3
Bit 4

Transmit error (should never occur)
Receive error (unexpected character or no char)
Time out error (expected .5 second response)
Parity error (sent/receive mismatch)
Enquiry error (no AJ71C24 response)

(Refer to AJ71C24 User Manual for details)

00H	Invalid access (made during run)
01H	Parity (sent/receive mismatch)
02H	Sum check (sent/receive mismatch)
03H	Protocol error (communication protocol not valid)
04H	Framing error (data/stop bit mismatch)
05H	Overrun error (previous data not yet received)
06H	Character area (mode set/character mismatch)
07H	Character error (illegal character/control code)
08H	PC access error (buffer to PC com err)

(Generated by AJ71C24 to PC CPU access)

10H	PC number error (station # doesn't exist, use FFH)
11H	Mode error (invalid request sent)
12H	SFU spec. error (no SFU at I/O location)
18H	Remote error (run/stop impossible)
21H	Memory access to SFU (control bus error/unit faulty)

24.5 COMMANDS

The 4800-E18 direct connect commands communicate between the OIL registers and the PLC's device memory bits, words, buffer memory, PC CPU, parameters, and Special Function Units. Bit/byte values are right justified in the 16-bit OIL registers. This section contains descriptions and examples of these commands.

24.5.1 GET PLC DEVICE BITS (GETDB)

Syntax:

GETDB stn,type,addr,lngth,dreg,sreg

where:

stn is the station number of the PLC (0-31)

type is the kind of device (bit value):

1 = X input	(000H-7FFH)
2 = Y output	(000H-7FFH)
3 = M relay	(0000-2047)
4 = L relay	(0000-2047)
5 = B relay	(000H-3FFH)
6 = F annunciator	(000-255)
7 = Special relay	(000-255)
8 = Timer contract	(000-255)
9 = Timer coil	(000-255)
10= Counter contact	(000-255)
11= Counter coil	(000-255)

addr is the device memory address (the range depends on type value above)

lngth is the length of the block (1-256)

dreg is the destination OIL register

sreg is the communication status register

This command reads PLC device memory data bits into OIL registers (1 bit per register).

Example:

GETDB 0,1,0,4,#20,#19

This command reads the first four X bits from PLC station 0 into OIL registers 20 to 23. Register 19 will contain the communication status.

24.5.2 GET PLC DEVICE WORD (GETDW)

Syntax:

GETDW stn,type,addr,lngth,dreg,sreg

where:

stn is the station number of the PLC (0-31)

type is the kind of device (1-11 = 16 bit values/word; 12-17 = 16-bit word value):

1 = X inputs word	(00H-7FH)
2 = Y outputs word	(00H-7FH)
3 = M relays word	(000-127)
4 = L relays word	(000-127)
5 = B relays word	(00H-3FH)
6 = F annunciators word	(00-15)
8 = Timer contracts word	(00-15)
9 = Timer coils word	(00-15)
10 = Counter contacts word	(00-15)
11 = Counter coils word	(00-15)
12 = Timer word	(000-255)
13 = Counter word	(000-255)
14 = Data register D word	(0000-1023)
15 = Link register W word	(000H-3FFH)
16 = File register R word	(0000-8191)
17 = Special register word	(000-255)

NOTE

Due to a bug in the Mitsubishi Aj71C24 unit, Xycom cannot support GETDW for memory type 7.

addr is the device memory address (the range depends on the type value above)

lngth is the length of the block (1-32 for types 1-11; 1-64 for types 12-17)

dreg is the destination OIL register

sreg is the communication status register

This command will read PLC device memory values and store them in a block of OIL registers (one word per register).

Example:

GETDW 1,1,0,1,#17,#18

This command will read the first 16 X bits from PLC station 1 into OIL register 17. Register 18 will contain the communication status.

24.5.3 GET BUFFER MEMORY VALUES (GETBUF)

Syntax:

GETBUF stn,type,addr,lnth,dreg,sreg

where:

stn is the station number of the PLC (0-31)

type is the type of communication
0 = Direct
1 = Indirect

addr is the memory device address
0-07FFH 16 bit values, for type = 0
1000H-1FFFH 8 bit values, for type = 1

lnth is the length of the block (1-64)

dreg is the destination OIL register

sreg is the communication status register

This command will read PLC buffer memory values and store them in a block of OIL registers (1 value per register).

Example:

GETBUF 2,0,&7FF,1,#17,#18

Using direct communication, this command will read the last 16-bit value from PLC station 2 to OIL register 17. Register 18 will contain the communication status.

24.5.4 GET PC CPU TYPE (GETCPU)

Syntax:

GETCPU stn,dreg,sreg

where:

stn is the station number of the PLC (0-31)

dreg is the destination OIL register

sreg is the communication status register

This command reads the PC CPU type (16-bit word value) from the PLC and stores the name (A1, A2, A3) in the OIL destination register.

Example:

GETCPU 3,#17,#18

This command reads the PC CPU type from PLC station 3 into register 17. Register 18 will contain the communication status.

24.5.5 GET PARAMETER MEMORY (GETPAR)

Syntax:

GETPAR stn,addr,lngth,dreg,sreg

where:

stn is the station number of the PLC (0-31)

addr is the memory device address (0-BFFH)

lngth is the length of the block (1-128)

dreg is the destination OIL register

sreg is the communication status register

This command will read 8-bit values from the 3 Kbytes of PC CPU parameter memory and store them in a block of OIL registers (1 byte per register).

Example:

GETPAR 4,0,128,#100,#228

This command will read 8-bit values from address 0 in PLC station 4 and store them in a 128 byte block in registers 100 to 227. Register 228 will contain the communication status.

24.5.6 GET SPECIAL FUNCTION UNIT BUFFER MEMORY (GETSFU)

Syntax:

GETSFU stn,type,addr,lngth,dreg,sreg

where:

- stn is the station number of the PLC (0-31)
- type is the Special Function Unit Number (0-255, byte)
- addr is the memory device address (depends on Special Function Unit)
- lngth is the length of block (1-128 bytes)
- dreg is the destination OIL register
- sreg is the communication status register

This command will read PLC Special Function Unit buffer memory and store values the in a block of OIL registers (1 byte per register). The format of the data values depends on the type and addr parameters. For more information, refer to the MELSEC-A User's Manual for the Computer Link Module AJ71C24.

Example:

GETSFU 0,0,0,1,#20,#18

This command will read the special function unit buffer memory of PLC station 0, unit 0, address 0, in one block to OIL register 20. Register 18 will contain the communication status.

24.5.7 PUT PLC DEVICE BITS (PUTDB)

Syntax:

PUTDB stn,type,addr,lngth,dreg,sreg

where:

stn is the station number of the PLC (0-31)

type is the kind of device (bit value):

1 = X input	(000H-7FFH)
2 = Y output	(000H-7FFH)
3 = M relay	(0000-2047)
4 = L relay	(0000-2047)
5 = B relay	(000H-3FFH)
6 = F annunciator	(000-255)
7 = Special relay*	(000-255)
8 = Timer contract	(000-255)
9 = Timer coil	(000-255)
10= Counter contact	(000-255)
11= Counter coil	(000-255)

NOTE

*Special relays have pre-defined meanings and should not be written to.

addr is the memory device address (the range depends on the type value above)

lngth is the length of block (1-160)

dreg is the OIL data register to read from

sreg is the communication status register

This command writes PLC device memory data bits from OIL registers (1 bit per register).

Example:

PUTDB 6,1,0,1,#17,#18

This example writes the first X bit from OIL register 17 to PLC station 6. Register 18 will contain the communication status.

24.5.8 PUT PLC DEVICE WORD (PUTDW)

Syntax:

PUTDW stn,type,addr,lngth,dreg,sreg

where:

stn is the station number of the PLC (0-31)

type is the kind of device (1-11 = 16-bit values/word; 12-17 = 16-bit word value):

1 = X inputs word	(00H-7FH)
2 = Y outputs word	(00H-7FH)
3 = M relays word	(000-127)
4 = L relays word	(000-127)
5 = B relays word	(00H-3FH)
6 = F annunciators word	(00-15)
7 = Special relays word*	(00-15)
8 = Timer contracts word	(00-15)
9 = Timer coils word	(00-15)
10 = Counter contacts word	(00-15)
11 = Counter coils word	(00-15)
12 = Timer word	(000-255)
13 = Counter word	(000-255)
14 = Data register D word	(0000-1023)
15 = Link register W word	(000H-3FFH)
16 = File register R word	(0000-8191)
17 = Special register word*	(000-255)

NOTE

*Special relays and special registers have pre-defined meanings and should not be written to.

addr is the memory device address (the range depends on the type value above)

lngth is the length of the block (1-10 for types 1-11; 1-64 for types 12-17)

dreg is the OIL data register to read from

sreg is the communication status register

This command writes PLC device memory word values from a block of 16-bit OIL registers.

Example:

PUTDW 1,1,0,1,#17,#18

This command will write the 16-bit data value from OIL register 17 to the first 16 X bits in PLC station 1. Register 18 will contain the communication status.

24.5.9 PUT BUFFER MEMORY VALUES (PUTBUF)

Syntax:

PUTBUF stn,type,addr,lngth,dreg,sreg

where:

stn is the station number of the PLC (0-31)

type is the type of communication
0 = Direct
1 = Indirect

addr is the memory device address
0-07FFH 16-bit values, for type=0
1000H-1FFFH 8-bit values, for type=1

lngth is the length of the block (1-64)

dreg is the OIL data register to read from

sreg is the communication status register

This command writes PLC buffer memory values from a block of OIL registers (1 value per register) to the target PLC.

Example:

PUTBUF 8,0,&7FF,1,#17,#18

Using direct communication, this command will write the last 16-bit value from a 1 Kbyte block in OIL register 17 to PLC station 8. Register 18 will contain the communication status.

24.5.10 PUT CPU RUN/STOP COMMAND (PUTCPU)

Syntax:

PUTCPU stn,com,sreg

where:

stn is the station number of the PLC (0-31)

com is the command
1 = Run
0 = Stop

sreg is the communication status register

This command writes a PC CPU run or stop command to the PLC station.

Example:

PUTCPU 9,0,#18

This command will stop PLC station 9. Register 18 will contain the communication status.

24.5.11 PUT PARAMETER MEMORY (PUTPAR)

Syntax:

PUTPAR stn,addr,lngth,dreg,sreg

where:

stn is the station number of the PLC (0-31)

addr is the memory device address (0-BFFH)

lngth is the length of the block (1-128)

dreg is the OIL data register to read from

sreg is the communication status register

This command will write 8-bit values to the 3 Kbytes of PC CPU parameter memory from a block of OIL registers (1 byte per register).

Example:

PUTPAR 10,0,128,#100,#228

This command will write a 128 byte block from registers 100-227 to PLC station 10's parameter memory starting at address 0. Register 228 will contain the communication status.

NOTE

The PC CPU must be stopped before writing to parameter memory.

24.5.12 PUT SPECIAL FUNCTION UNIT BUFFER MEMORY (PUTSFU)

Syntax:

PUTSFU stn,type,addr,lngth,dreg,sreg

where:

- stn is the station number of the PLC (0-31)
- type is the type ("Special Function Unit Number", 0-255, byte)
- addr is the memory device address (depends on "Special Function Unit")
- lngth is the length of the block (1-128 bytes)
- dreg is the OIL data register to read from
- sreg is the communication status register

This command writes PLC Special Function Unit buffer memory with values from the block of OIL registers (1 byte per register). The format of the data values depends on the type and addr parameters. For more information, refer to the User Manual for the MELSEC-A Computer Link Module AJ71C24.

Example:

PUTSFU 0,0,0,2,#20,#18

This command will write the special function unit buffer memory value in OIL register 20, to PLC station 0, unit 0, address 0 in a 2 Byte block. Register 18 will contain the communication status.

24.6 PROGRAM EXAMPLE

The following is a sample OIL program which reads and writes PLC values, using each of the 4800-E18 direct connect user interface commands. (This assumes the OIL registers used are loaded with the PLC values to write.)

```
PUTDB 0,1,&3,1,#10,#70
" Data sent & cs = " : tr,#10,sc(hhhh ) : tr,#70,sc(hhhh) : nl
GETDB 0,1,&3,1,#11,#70
" Received data & cs = " : tr,#11,sc(hhhh ) : tr,#70,sc(hhhh) : nl

PUTDW 0,1,0,1,#20,#70
" Data sent & cs = " : tr,#20,sc(hhhh ) : tr,#70,sc(hhhh) : nl
GETDW 0,1,0,1,#21,#70
" Received data & cs = " : tr,#21,sc(hhhh ) : tr,#70,sc(hhhh) : nl

PUTBUF 0,0,&7ff,1,#30,#70
" Data sent & cs = " : tr,#30,sc(hhhh ) : tr,#70,sc(hhhh) : nl
GETBUF 0,0,&7ff,1,#31,#70
" Received data & cs = " : tr,#31,sc(hhhh ) : tr,#70,sc(hhhh) : nl

PUTCPU 0,0,#70
" STOP command cs = " : tr,#70,sc(hhhh)
" PLC STOP Press any key to continue : " : tr,kb(1),#90 : nl

PUTCPU 0,1,#70
" RUN command cs = " : tr,#70,sc(hhhh)
" PLC RUN Press any key to continue : " : tr,kb(1),#90 : nl

GETCPU 0,#41,#70
" MELSEC PLC CPU type & CS = " : tr,#41,sc(cc ) : tr,#70,sc(hhhh ) : nl

PUTPAR 0,0,1,#50,#70
" Data sent & cs = " : tr,#50,sc(hh ) : tr,#70,sc(hhhh) : nl
GETPAR 0,0,1,#51,#70
" Received data & cs = " : tr,#51,sc(hh ) : tr,#70,sc(hhhh) : nl

PUTSFU 0,8,&80,1,#60,#70
" Data sent & cs = " : tr,#60,sc(hh ) : tr,#70,sc(hhhh) : nl
GETSFU 0,8,&80,1,#61,#70
" Received data & cs = " : tr,#61,sc(hh ) : tr,#70,sc(hhhh)

stop
```


Chapter 25

4800-E19 Texas Instruments Series 305 PLC Interface

25.1 INTRODUCTION

This chapter contains information specific to the 4800-E19 Texas Instruments Series 305 PLC interface. To communicate with the TI 305, this module uses a full-duplex, asynchronous, point-to-point protocol that closely conforms to ANSI X3.28.

The specific OIL commands that allow communication are described in Section 25.6 and listed below for reference:

GETR	Get register
GETIOS	Get I/O status
PUTR	Put register
PUTIOS	Put I/O status

25.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC via a non-standard RS-232C cable you must fabricate. Figure 25-1 shows the connections you must make between the Expansion Card port and the PLC.

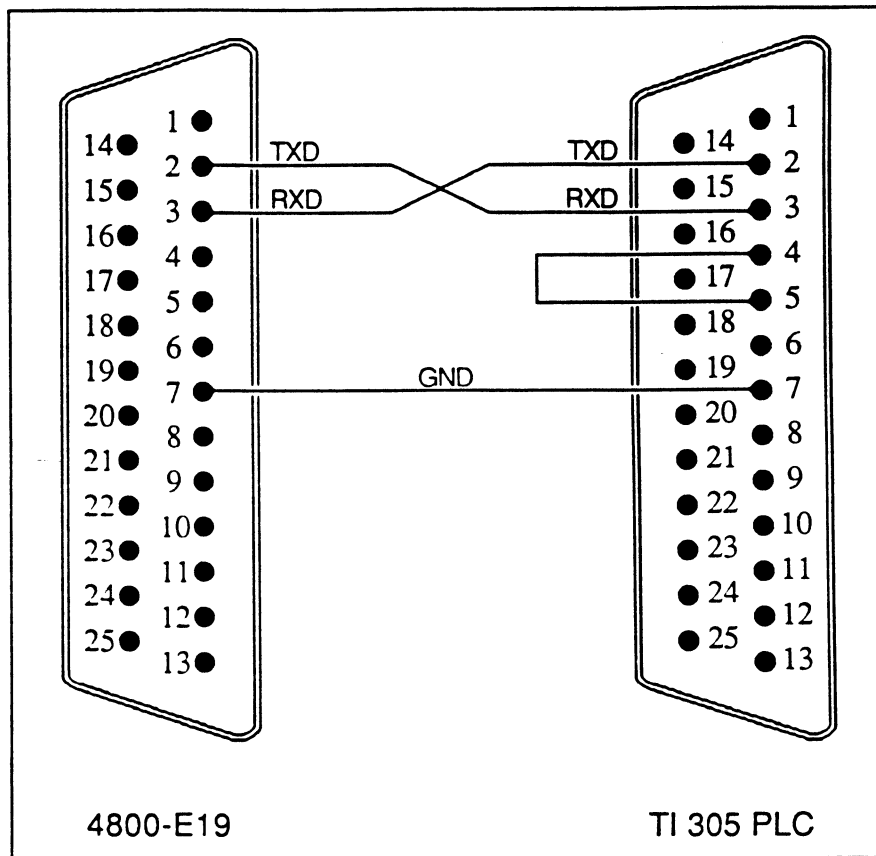


Figure 25-1. Connections

25.3 CONFIGURATION MENU

The Configuration Menu is called up from the Main Menu (see Chapter 3) and is shown in Figure 25-2 below:

```
-- Texas Instruments PLC Port Configuration Menu --  
  
6      Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600 7=19.2 Kbyte  
01     Parity - 0=No Parity 1=Odd Parity  
01     Source ID# (1-90)  
01     Timeout Value (1-20 seconds)  
  
Use <UP ARROW>, <DOWN ARROW>, <LEFT ARROW>, <RIGHT ARROW>.  
Use values 0 through 9.  
"C" for next configuration menu, <RET> or <ENTER> to quit.
```

Figure 25-2. Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Parity. This option sets the parity used in communication to the Texas Instruments 305 PLC to either none or odd. The type selected should match the other communication device(s).

Source ID#. The source ID number setting in the TI-305 Configuration Menu **must** be identical to the station address of the TI PLC to which the terminal is directly connected. Refer to the Texas Instruments user reference material for information on obtaining the station address.

Timeout Value. The timeout value determines how many seconds (from 1 to 20) the terminal waits for a response before signaling a timeout.

25.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figure below illustrates the configuration of the test plugs (including the optional RS-422 port).

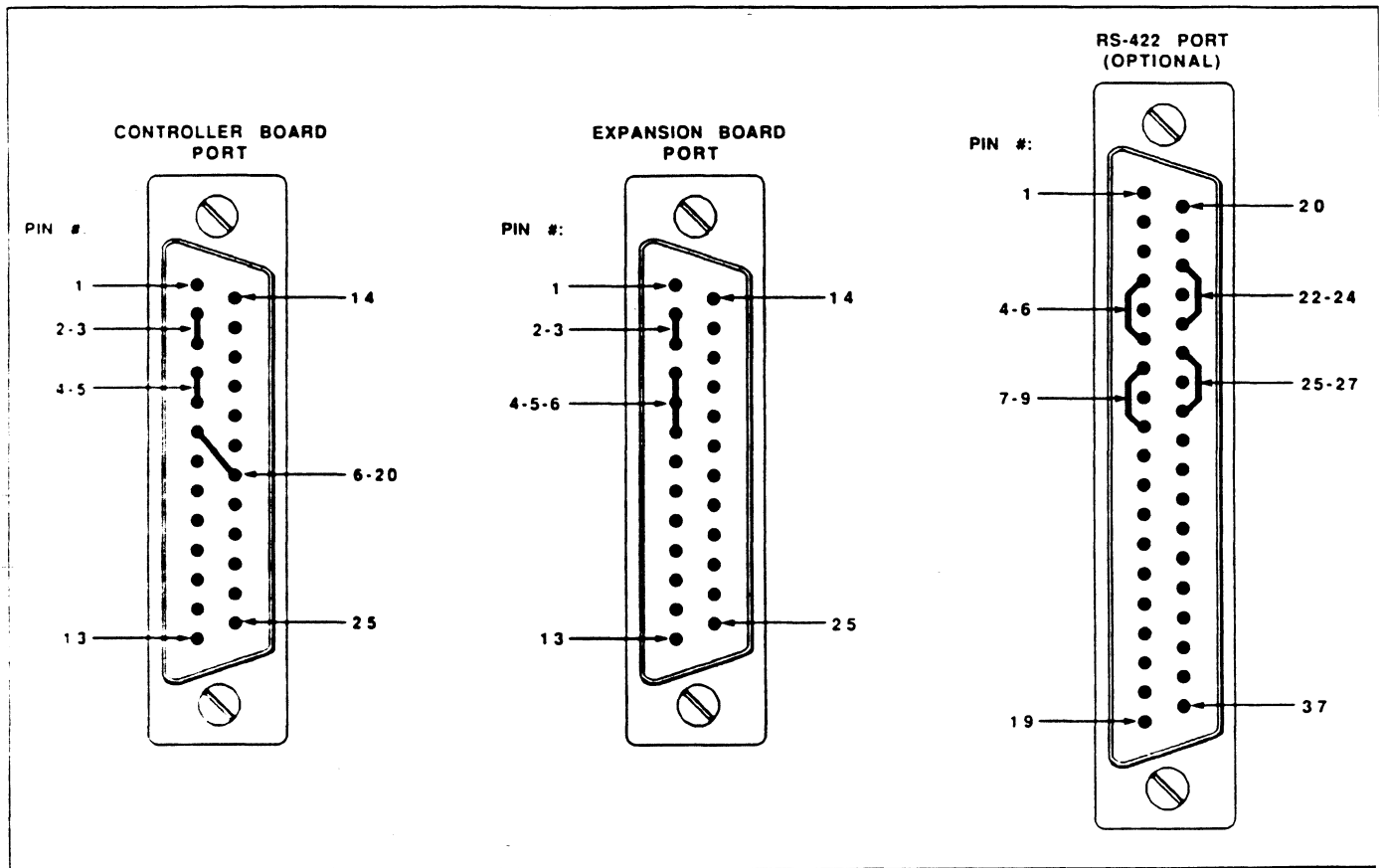


Figure 25-3. Serial Port Test Plugs

25.5 COMMUNICATION STATUS REGISTER

The communication status register contains the status about the message that was just transferred. The Most Significant Byte (MSB) is not used. The Least Significant Byte (LSB) contains information about the message transfer status. A zero value in the entire register indicates a successful transfer.

If the bit noted is set, the condition indicated is true.

Status Register:

MSB: NOT USED

LSB: Message Transfer Status

Bit 0 = Transmit Error
Bit 1 = Receive Error
Bit 2 = Timeout
Bit 3 = Parity Error
Bit 4 = Enquiry Error
Bits 5-7 = NOT USED

25.6 COMMANDS

This section describes the commands that are specific to the 4800-E19: Texas Instruments Series 305.

25.6.1 Get Register (GETR)

Syntax:

```
GETR stn,reg,lngh,dreg,sreg
```

where

stn is the station address of the target PLC

reg is the register in the target PLC from which the data is to be read

lngh is the number of words to read

dreg is the destination register number to store data

sreg is the communication status register

The command GETR reads data from a timer/counter accumulator or registers in the PLC. Refer to Section 25.5 for the communication status register information.

Example:

```
GETR 10,20,10,#30,#20
```

This command reads 10 timer/counter accumulator values starting at timer/counter 20 of the target device with station address 10. The data returned is placed into registers 30 through 39. Register 20 contains the communication status.

25.6.2 Get I/O Status (GETIOS)

Syntax:

GETIOS stn,addr,lngth,dreg,sreg

where

stn is the station address of the target PLC

addr is the starting address to read

lngth is the number of bytes to read, starting at addr

dreg is the stination register number to store data

sreg is the communication status register

The command GETIOS reads the I/O bits (input/output, internal relay, shift register, and timer/counter) from the PLC. Refer to Section 25.5 for the communication status register information.

Example:

GETIOS 1,1,2,#100,#70

This command reads I/O address 0 to 7 and 10 to 17 from the TI 305 and puts the bit patterns into the lower bytes of registers 100 and 101. The status information is stored in register 70.

25.6.3 Put Register (PUTR)

Syntax:

PUTR stn,reg,lngh,srcreg,sreg

where

stn is the station address of the target PLC

reg is the register in the target PLC from which the data is to be written

lngh is the number of words to write

srcreg is the source register number to send data from

sreg is the communication status register

The command PUTR sends data to a timer/counter accumulator or register in the PLC. Refer to Section 25.5 for the communication status register information.

Example:

PUTR 15,33,13,#30,#20

This command writes 13 words of data starting at timer/counter accumulator 33 of the target device with station address 15. The data is sent from into registers 30 through 42. Register 20 contains the communication status.

25.6.3 Put I/O Status (PUTIOS)

Syntax:

PUTIOS stn,addr,lngth,srcreg,sreg

where

stn	is the station address of the target PLC
addr	is the starting address of input/output status to write
lngth	is the number of bytes to write, starting at addr
srcreg	is the source register number to write data from
sreg	is the communication status register

The command PUTIOS writes a byte of I/O (input/output, internal relay, shift register) information to the PLC. Refer to Section 25.5 for the communication status register information.

Example:

PUTIOS 1,1,2,#100,#40

This command writes the bit pattern from the lower byte of register 100 to I/O addresses 0 to 7, and the bit pattern from register 101 to I/O addresses 10 to 17. Register 40 contains the communication status.

Chapter 26

4800-E20 GE Series 90 PLC

26.1 INTRODUCTION

This chapter is reserved for the future release of the 4800-E20 GE Series 90 PLC direct connect module.

Chapter 27

4800-E21 GEM 80 Interface

27.1 INTRODUCTION

The 4800-E21 Expansion Module allows the user to access data tables on GEM 80 PLC's via Xycom's OIL programming language.

The specific commands that allow communications are listed in Section 27.6, where they are discussed in detail with examples. These commands are:

GETWD
GETBLK
PUTWD
PUTBLK
SWAPJK

The communication protocol between the 4800-E21 and the GEM 80 PLC (point-to-point or multidrop) is ESP, or Extended Simple Protocol. ESP uses asynchronous communication, with each character transmitted as 1 start bit, 8 data bits, and 1 stop bit.

ESP requires that all communication be between a control station and one or more tributary stations. A tributary may only transmit after it has received a message from the control station and this is taken as an invitation to reply to the control station *only*, not to another tributary.

27.2 CABLING THE TERMINAL TO THE PLC

The Expansion Card is attached to the PLC and/or network via a non-standard RS-232 cable you must fabricate. Figure 27-1 illustrates the connections that must be made between the Expansion Card port and the PLC. The distance between the two devices must be limited to 50 feet or less.

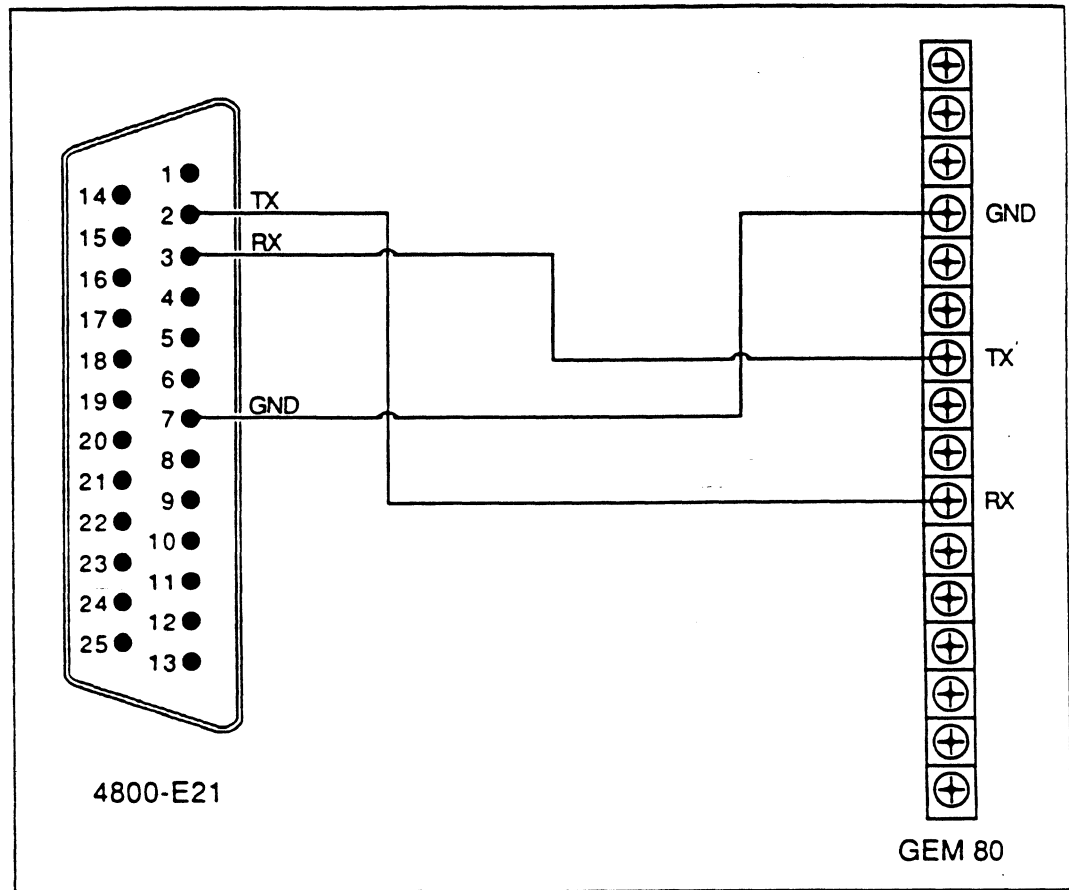


Figure 27-1. Connections

NOTE

You must use shielded cable and tie the shield to chassis ground.

27.3 CONFIGURATION MENU

The configuration menus are called up from the Main Menu (discussed in Chapter 3). Figure 27-2 shows the GEM 80 Configuration Menu.

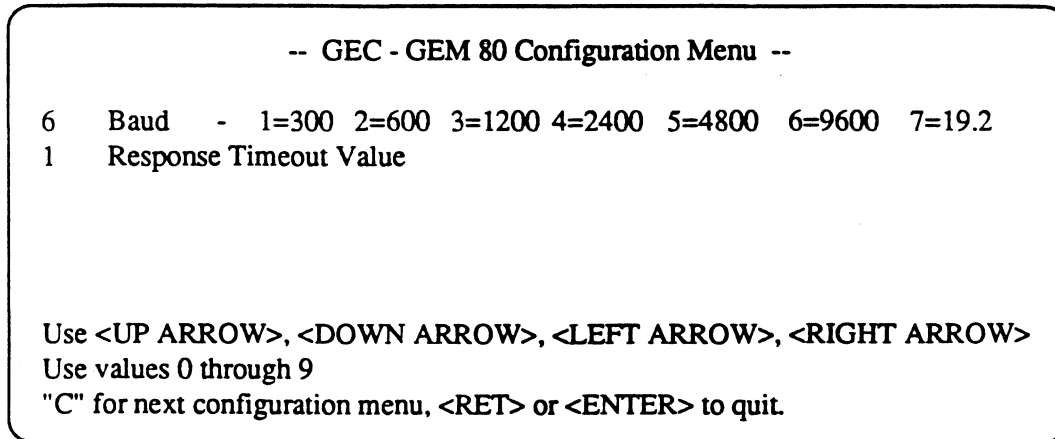


Figure 27-2. Configuration Menu

Baud. The baud rate of the channel should be set to match that of the PLC.

Response Timeout. The timeout option defines how long the terminal will wait for a response before it signals a timeout. The value is entered in seconds, with a range of 1-5.

27.4 SERIAL LOOP BACK TEST

The serial ports on the terminal can be tested by selecting item #4 (Serial Loop Back) from the Diagnostic Menu. This test requires certain signals to be "looped back" to the terminal for signal verification. The recommended means for looping signals is to construct a loop-back connector using a D-type connector of the same size as the port. The figures below illustrate the configuration of the test plugs.

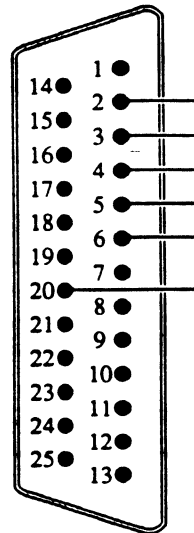


Figure 27-3. Controller Board Test Plug

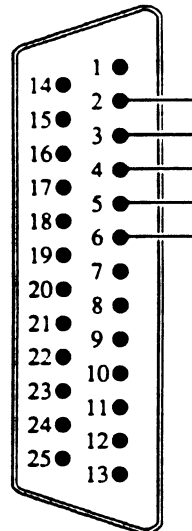


Figure 27-4. Expansion Board Test Plug

27.5 COMMUNICATION STATUS REGISTER

The Communication Status Register contains the status about the message that was just transferred. The Most Significant Byte is not used. The Least Significant Byte contains information about message transfer status on the terminal end.

If the bit noted is set, the condition indicated is true.

NOTE

When bit 2 is set in the communication status register, it indicates that the user tried to do something the PLC cannot perform, such as reading an invalid table location; it is NOT a communication error.

Status Register:

MSB - Not used

LSB - Bit 0 = Receive error
Bit 1 = Timeout error
Bit 2 = Received ESP ?
Bit 3 = CRC Compare error
Bit 4 = Received NAK
Bits 5-8 = Not used

27.6 COMMANDS

This section describes the commands that are specific to the 4800-E21 GEM 80 PLC network interface.

NOTE

All tables can be read from, but only E, G, H, N, O, Q, R, U, and W can be written to.

Different tables contain a different number of locations, and one table may have a different number of locations in different PLCs. You must be aware of the table configurations for the particular PLC you are using. If you attempt to read from a location which does not exist in the table on your PLC, the system will flag an error (ESP ?) in the communication status register.

For information on table configurations, consult your GEM 80 documentation.

27.6.1 Read a Word (GETWD)

Syntax:

```
GETWD trib,"tabl".loc,dreg,csreg
```

where:

- trib - Number of tributary station being addressed.
- tabl - The table letter in quotes.
- loc - Location within table specified above, 0 - 9999
(Depends on how tables are configured).
- dreg - Destination OIL register to store recieved data in.
- csreg - OIL register to store communication status in.

The GETWD command reads one word out of a selected data table into a specified OIL register.

Example: GETWD 5,"A",0,#30,#20

This command would read the word located at table A location 0 of the fifth tributary station into OIL register number 30 and the communication status would be returned into OIL register number 20.

27.6.2 Read a Block (GETBLK)

Syntax:

```
GETBLK trib,"tabl",loc,num,dreg,csreg
```

where:

- dest - Number of tributary station being addressed.
- tabl - The table letter in quotes.
- loc - Location of start of block within table specified above, 0 - 9999
(Depends on how tables are configured).
- num - two digit number representing the quantity of locations to
be read (in range 1-32).
- dreg - Destination OIL register to store first word of block of recieved data in.
- csreg - OIL register to store communication status in.

This command reads the contents of a block of contiguous table locations, starting at a given table location, into a block of OIL registers, also starting at a given location.

Example: GETBLK 3,"K",10,10,#45,#20

This command would read 10 16 bit words from K10 through K19 from the third tributary station into OIL registers 45 through 54 and store the communication status in OIL register 20.

27.6.3 Write a Word (PUTWD)

Syntax:

```
PUTWD trib,"tabl",loc,sreg,csreg
```

where:

- trib - Number of tributary station being addressed.
- tabl - Table letter in quotes.
- loc - Location within table specified above, 0 - 9999
(Depends on how tables are configured).
- sreg - Source OIL register containing word to be sent.
- csreg - OIL register containing communication status.

This command writes one word from a selected OIL register into a selected table location.

Example: PUTWD 6,"N",20,#99,#20

This command would write the contents of OIL register 99 into table N location 20, of the sixth tributary station the communication status would be stored in OIL register 20.

27.6.4 Write a Block (PUTBLK)

Syntax:

```
PUTBLK trib,"tabl",loc,num,sreg,csreg
```

where:

- trib - Number of tributary station being addressed.
- tabl - Table letter in quotes.
- loc - Location of start of block within the table specified above, 0 - 9999
(Depends on how tables are configured).
- num - The number of locations to be written to, i.e. the number of OIL registers being sent (in range 1 - 32).
- sreg - Source OIL register which marks the start of the block being sent.
- csreg - OIL register containing communication status.

This command writes data stored in selected OIL registers to contiguous locations in the selected table.

Example: PUTBLK 2,"G",2,20,#50,#20

This command writes a block of data contained in OIL registers 50 through 69 into table G locations 2 through 21 of the second station tributary. The communication status is stored in OIL register 20.

27.6.5 Exchange Blocks (SWAPJK)

Syntax:

SWAPJK trib,dreg,num,sreg,csreg

where:

- trib - Number of the tributary station being addressed.
- dreg - OIL register into which the first word of data is returned (Dummy K table, returns 1 - 32 words, depending on how the PLC is configured).
- num - Number of words being sent to the PLC.
- sreg - OIL register containing the beginning of the data to be sent (Dummy J table).
- csreg - OIL register containing communication status.

This command enables J-K table exchange between the Xycom terminal and an older GEM 80 PLC (which only support J-K exchange).

Example: SWAPJK 10,#30,15,#60,#20

This command would write data from OIL registers 60 through 74 to the tributary and read data sent back (will be same size as that sent) into OIL registers 30 through 44. The communicationstatus would be stored in OIL register 20.

Appendix A

SAVING/LOADING/VERIFYING SCREEN PROGRAMS

A.1 INTRODUCTION

The following instructions describe, in detail, how to save, reload and verify stored screens using the IBM PC and the STR-LINK III.

If the expansion board used requires a communication adapter, it may be necessary to tie certain pins together to enable the program transfers described in this chapter. See the appropriate chapter for details.

A.2 IBM PC

When using the IBM PC, all screens will be stored in standard ASCII format.

A.2.1 Saving Screen Programs

1. Connect the terminal serial port (or option port if a communication adapter is used) 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:

<u>PC/AT PIN #</u>	<u>PC/XT PIN #</u>	<u>SIGNAL NAME</u>	<u>TERMINAL PIN #</u>
1	1	Cable Shield	N/C
3	2	Transmit Data	3
2	3	Receive Data	2
5	7	Logic Ground	7

NOTE
Pins 4, 5, and 6 must be shorted together on the terminal side of the cable.

Load and configure your particular ASCII communication program for the proper communication parameters.

2. Configure the terminal for the following parameters:

- 8 Data Bits
- Full Duplex
- Handshaking (XON, XOFF)
- Baud Rate (to match communications package)
- Parity (to match communications package)

3. On the terminal, choose the Backup/Restore function (Main Menu item #3), then choose the Backup function (Backup/Restore menu item #1).
4. The terminal 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 terminal prompts for which of the 255 programs 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.

During the save procedure, the IBM PC displays the data it receives. When the display stops, enter screen program backup is complete.

A.2.2 Restoring Screen Programs

Perform steps 1 and 2 from above.

3. On the terminal, choose the Backup/Restore function (Main Menu item #3), then choose the Restore function (Backup/Restore menu item #2).
4. Type the command that displays the prompt to enter the name of the file that contains the screen to load. Then enter the file name.

The screen(s) will be sent to the terminal and stored as the screen number you gave them when they were saved. This completes the program restoration.

A.2.3 Verifying Screen Programs

Perform steps 1 and 2 from above.

3. On the terminal, choose the Backup/Restore function (Main Menu item #3), then choose the Verify function (Backup/Restore menu item #3).
4. Type the command that displays the prompt to enter the name of the file containing the screen you wish to verify. Then enter the file name.

The screen(s) will be sent to the terminal and verified. If an error occurs during verification, the terminal immediately displays a "Verification failed" message.

A.3 STR-LINK III

A.3.1 Saving Screen Programs

1. Connect the terminal to the "Data Terminal Interface" connector on the STR-LINK III to the terminal serial port (or option port if a communication adapter is used) with a 25-pin ribbon cable.
2. Set the switches on the STR-LINK III as follows:

<u>SWITCH</u>	<u>SETTING</u>	<u>FUNCTION</u>
1	CLOSED	Normal operation
2	CLOSED	Connect signal and earth grounds
3	CLOSED	Disable retries
4	OPEN	Disable control character decoding
5	CLOSED	Enable control lines
6	CLOSED	Disable TTY
7	CLOSED	Disable current loop
8	CLOSED	Disable data echoing
9	CLOSED	Off line
10	CLOSED	Full-duplex

3. Set the baud rate switch on the STR-LINK III to the desired baud rate. The screen transfer will work at any of the six available baud rates.
4. Choose a track on the STR-LINK III. The entire screen memory of a terminal will fit on a single track.

5. Turn on the STR-LINK III.
6. Load a digital cartridge and wait for the Stop/Rewind light to go out.
7. Turn on the terminal.
8. Set the terminal configuration options as follows:

Desired baud rate
Full-duplex
8 data bits
Parity disabled
Handshaking disabled

Other terminal options will not affect the screen transfer.

9. On the terminal, choose the BACKUP/RESTORE function (Main Menu item #3), then choose the Backup function (BACKUP/RESTORE Menu item #1).
10. The terminal will prompt for a backup of programs, data registers, or both programs and data registers (all). Press the Remote button on the STR-LINK III.
11. If backup of programs was selected in step 10, the terminal will prompt for which of the 255 programs 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 in the lower right-hand corner of the screen.
12. When the terminal indicates screen transmission is done, press the Stop/Rewind button on the STR-LINK III.

This completes the screen program backup.

A.3.2 Restoring Screen Programs

Perform steps 1 through 8 as detailed above, but load a digital cartridge which has already had screen data recorded on it.

9. On the terminal, choose the BACKUP/RESTORE function (Main Menu item #3), then choose the Restore function (BACKUP/RESTORE Menu item #2).
10. Press the Remote button on the STR-LINK III.

11. When the DATA OUT light on the STR-LINK III goes out, type F5 or CLR on the terminal.
12. Press the Stop/Rewind button on the STR-LINK III.

This completes the program restoration.

A.3.3 Verifying Screen Programs

Perform steps 1 through 8 as detailed above.

9. On the terminal, choose the BACKUP/RESTORE function (Main Menu item #3), then choose the Verify function (BACKUP/RESTORE Menu item #3).
10. Press Remote button on the STR-LINK III.
11. If an error occurs during verification, the terminal will immediately display a "Verification failed" message.
12. If no errors occurred, press the F5 or CLR key when the DATA OUT light on the STR-LINK III goes out. Then press the Stop/Rewind button.

Appendix B

KEYPAD KEY CODES

B.1 KEYPAD CODES

The following table lists the keypad key codes for the two keypad versions found on XYCOM Industrial Terminals.

3 x 10 Keypad				4 x 7 Keypad			
KEY	ASCII	DEC	HEX	KEY	ASCII	DEC	HEX
F1	G	71	47	A	A	65	41
F2	H	72	48	PAUSE	P	80	50
F3	I	73	49	↑	DC1	17	11
F4	J	74	4A	CLEAR	<None>	132	84
F5	<None>	132	84	B	B	66	42
F6	<None>	133	85	←	DC2	18	12
A	A	65	41	↓	DC4	20	14
B	B	66	42	→	DC3	19	13
C	C	67	43	C	C	67	43
D	D	68	44	7	7	55	37
E	E	69	45	8	8	56	38
F	F	70	46	9	9	57	39
7	7	55	37	D	D	68	44
8	8	56	38	4	4	52	34
9	9	57	39	5	5	53	35
4	4	52	34	6	6	54	36
5	5	53	35	E	E	69	45
6	6	54	36	1	1	49	31
1	1	49	31	2	2	50	32
2	2	50	32	3	3	51	33
3	3	51	33	F	F	70	46
0	0	48	30	0	0	48	30
↑	DC1	17	11	.	.	46	2E
.	.	46	2E	ENTER	<CR>	13	0D
←	DC2	18	12	I	G	71	47
↓	DC4	20	14	II	H	72	48
→	DC3	19	13	III	I	73	49
ENTER	<CR>	13	0D	IV	J	74	4A

Appendix C
OIL COMMANDS

Table C-1 OIL Commands by Section

COMMAND	SYNTAX*	SECTION
Screen Text	"{character}"	4.6.2
Register Value Display	<data register>	4.6.3
Position	@[<xcoord>],[<ycoord>]	4.6.4
Label	%<label>	4.6.5
Add to Register	ADD,<numerical argument>,<data register>	4.6.6
AND	AND,<data register>,<numerical argument>	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,<data register>,<bit>	4.6.12
Clear Screen	CS	4.6.13
Clear Window	CW	4.6.14
Down	D[,<numerical argument>]	4.6.15
Decrement Register	DEC,<data register>	4.6.16
Divide Register	DIV,<data register>,<numerical argument>	4.6.17
Execute Sub-Program Block	ES,<numerical argument>	4.6.18
Execute Program Block	EX,<numerical argument>	4.6.19
Exit	EXIT	4.6.20
Draw Filled Box	FBX,<bval>,[<xstart>],[<ystart>],[<xend>],[<yend>]	4.6.21
Go	GO,<label>	4.6.25
Horizontal Bar Left	HBL,[<xstart>],[<ystart>],<length>,<max length>	4.6.26
Horizontal Bar Right	HBR,[<xstart>],[<ystart>],<length>,<max length>	4.6.27
Draw Horizontal Line	HL,<lval>,[<xstart>],[<ystart>],<hlength>	4.6.28

Table C-1 OIL Commands by Section cont.

COMMAND	SYNTAX*	SECTION
If	IF,<condition>:<command>	4.6.29
If Bit	IFB,<data register>,<bit>:<command>	4.6.30
Increment Register	INC,<data register>	4.6.31
KEXIT	KEXIT	4.6.32
Keypad Status	KS,<data register>	4.6.33
Left	L[,<numerical argument>]	4.6.34
Remainder (Modulus)	MOD,<data register>,<numerical argument>	4.6.35
Multiply Register	MUL,<data register>,<numerical argument>	4.6.36
New Line	NL[,<numerical argument>]	4.6.37
NOT	NOT,<data register>	4.6.38
ONKEY	ONKEY,<numerical argument>	4.6.39
OR	OR,<data register>,<numerical argument>	4.6.40
Plot	PLOT,<xpoint>,<ypoint>	4.6.41
Pause	PS,<numerical argument>	4.6.42
Right	R[,<numerical argument>]	4.6.46
Reset Attributes	RE,{<attribute>}	4.6.47
Restore Attributes	RSTA,<data register>	4.6.48
Save Attributes	SAVA,<data register>	4.6.49
Set Background Color**	SBC,<color>	4.6.50
Set Attributes	SE,{<attribute>}	4.6.51
Set Bit	SETB,<data register>,<bit>	4.6.52
Set Foreground Color**	SFC,<color>	4.6.53
Stop	STOP	4.6.54
Subtract from Register	SUB,<numerical argument>,<data register>	4.6.55
Transfer	TR,<source>,<destination>	4.6.56
Up	U[,<numerical argument>]	4.6.57
Unplot	UNPLOT,<xpoint>,<ypoint>	4.6.58
Vertical Bar Up	VB VBU,[<xstart>],[<ystart>],<new height>,<max height>	4.6.59
Vertical Bar Down	VBD,[<xstart>],[<ystart>],<new height>,<max height>	4.6.60
Draw Vertical Line	VL,<lval>,[<xstart>],[<ystart>],<vlength>	4.6.61
XOR	XOR,<data register>,<numerical argument>	4.6.62

*See the next page for the meaning of the symbols used here.

**Color terminals only.

4.2 COMMAND LIST

Table C-2 lists all the available OIL commands and their formats. The OIL commands and Section numbers are listed in table C-2.

Table C-2 OIL Commands by Function

COMMAND	SYNTAX*	SECTION
Screen Text	"{character}"	4.6.2
Register Value Display	<data register>	4.6.3
select character size and attributes		
attributes (i.e., blinking or reverse video)		
Set Attributes	SE,<attribute>	4.6.49
Reset Attributes	RE,<attribute>	4.6.45
Save Attributes	SAVA,<data register>	4.6.47
Restore Attributes	RSTA,<data register>	4.6.46
color commands (Color Terminal only)		
Set Foreground Color**	SFC,<color>	4.6.51
Set Background Color**	SBC,<color>	4.6.48
move the cursor		
Up	U[,<numerical argument>]	4.6.55
Down	D[,<numerical argument>]	4.6.15
Left	L[,<numerical argument>]	4.6.33
Right	R[,<numerical argument>]	4.6.44
Position	@[<xcoord>],[<ycoord>]	4.6.4
New Line	NL[,<numerical argument>]	4.6.36
draw figures and lines		
Draw Box	BX,<bval>,[<xstart>],[<ystart>],[<xend>],[<yend>]	4.6.9
Draw Filled Box	FBX,<bval>,[<xstart>],[<ystart>],[<xend>],[<yend>]	4.6.21
Draw Vertical Line	VL,<lval>,[<xstart>],[<ystart>],[<vlength>]	4.6.59
Draw Horizontal Line	HL,<lval>,[<xstart>],[<ystart>],[<hlength>]	4.6.28
Vertical Bar Up	VB VBU,[<xstart>],[<ystart>],[<new height>],[<max height>]	4.6.57
Vertical Bar Down	VBD,[<xstart>],[<ystart>],[<new height>],[<max height>]	4.6.58
Horizontal Bar Right	HBR,[<xstart>],[<ystart>],[<new length>],[<max length>]	4.6.27
Horizontal Bar Left	HBL,[<xstart>],[<ystart>],[<new length>],[<max length>]	4.6.26
Plot	PLOT,<xpoint>,<ypoint>	4.6.39
Unplot	UNPLOT,<xpoint>,<ypoint>	4.6.56

Table C-2 OIL Commands by Function cont.

COMMAND	SYNTAX*	SECTION
move data between the keyboard, data registers, and host computer		
Transfer	TR,<source>,<destination>	4.6.54
conditional and looping commands		
If	IF,<condition>:<command>	4.6.29
If Bit	IFB,<data register>,<bit>:<command>	4.6.30
Go	GO,<label>	4.6.25
Label	%<label>	4.6.5
set and clear bits of a data register		
Set Bit	SETB,<data register>,<bit>	4.6.50
Clear Bit	CLRB,<data register>,<bit>	4.6.12
arithmetic operations		
Increment Register	INC,<data register>	4.6.31
Decrement Register	DEC,<data register>	4.6.16
Add to Register	ADD,<numerical argument>,<data register>	4.6.6
Subtract from Register	SUB,<numerical argument>,<data register>	4.6.53
Multiply Register	MUL,<data register>,<numerical argument>	4.6.35
Divide Register	DIV,<data register>,<numerical argument>	4.6.17
Remainder (Modulus)	MOD,<data register>,<numerical argument>	4.6.34
logical operations		
AND	AND,<data register>,<numerical argument>	4.6.7
XOR	XOR,<data register>,<numerical argument>	4.6.60
OR	OR,<data register>,<numerical argument>	4.6.38
NOT	NOT,<data register>	4.6.37
program nesting		
Execute Program Block	EX,<numerical argument>	4.6.19
Execute Sub-Program Block	ES,<numerical argument>	4.6.18
Exit	EXIT	4.6.20
Stop	STOP	4.6.52

Table C-2 OIL Commands by Function cont.

COMMAND	SYNTAX*	SECTION
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
Keypad Status	KS,<data register>	4.6.32
Pause	PS,<numerical argument>	4.6.40
ONKEY	ONKEY,<numerical argument>	4.6.39
KEXIT	KEXIT	4.6.32

*See below for the meaning of the symbols used here.

**Color terminals only.

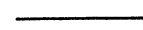

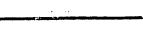


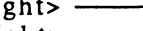
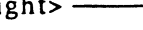
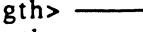



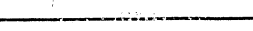

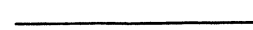
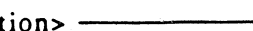
KEY

<u>SYMBOL</u>	<u>MEANING</u>
[] _____	Optional argument
{ } _____	One or more occurrences of this argument can be included
_____	Choice of either argument
<character> _____	Any single character (e.g., N, m, \$)
<data register> _____	#n or \$n, where n=1 through 500
<bit> _____	A specific bit within a register (0-15)
<numerical argument> _____	A number (0-65535), data register, or text argument
<text argument> _____	One or two characters enclosed in double quotes (e.g., "A" or "ra")
<xcoord> _____	Column coordinate (0 - 79)
<xstart> _____	
<xend> _____	

KEY cont.

SYMBOL

MEANING

<ycoord>		Row coordinate (0 - 23)
<ystart>		
<yend>		
<vlength>		Vertical length of graphic (0 - 24)
<hlength>		Horizontal length of graphic (0 - 80)
<new height>		Height of graphic (0 - 255, 0 - 240 for color)
<max height>		
<new length>		Length of graphic (0 - 255)
<max length>		
<xpoint>		Column coordinate (0 - 159)
<ypoint>		Row coordinate (0 - 71, 0 - 47 for color)
<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>
<destination>		Destination for data in a transfer. Any of the following: SC (screen output) SO (serial output) NO (dummy output) \$n <data register>

KEY cont.

<u>SYMBOL</u>	<u>MEANING</u>
<attribute>	Character attribute. Any of the following: QS (quad-size) DS (double size) RV (reverse video) monochrome only HI (highlight) monochrome only 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)
<condition>	A logical function inserted between two numerical arguments It can be: <, >, <>, =, =>, >=, <=, or =<
<label>	A sequence of alphanumeric characters
<color>	A field color. It can be: BLK, BLU, GRN, CYN, RED, MAG, YEL, or WHT or 0, 1, 2, 3, 4, 5, 6, or 7
<device>	A device used to handle data. It can be: KB (keyboard or keypad) SI (serial input) SO (serial output)
<addr>	Address from which data is to be read
<dest>	Address of remote PLC device.
<lngth>	Number of words (16-bit values) to read, starting at addr.
<dreg>	Destination Register number to store received data into.
<creg>	Communication Status Register.
<sreg>	Source Register number to send data from.

