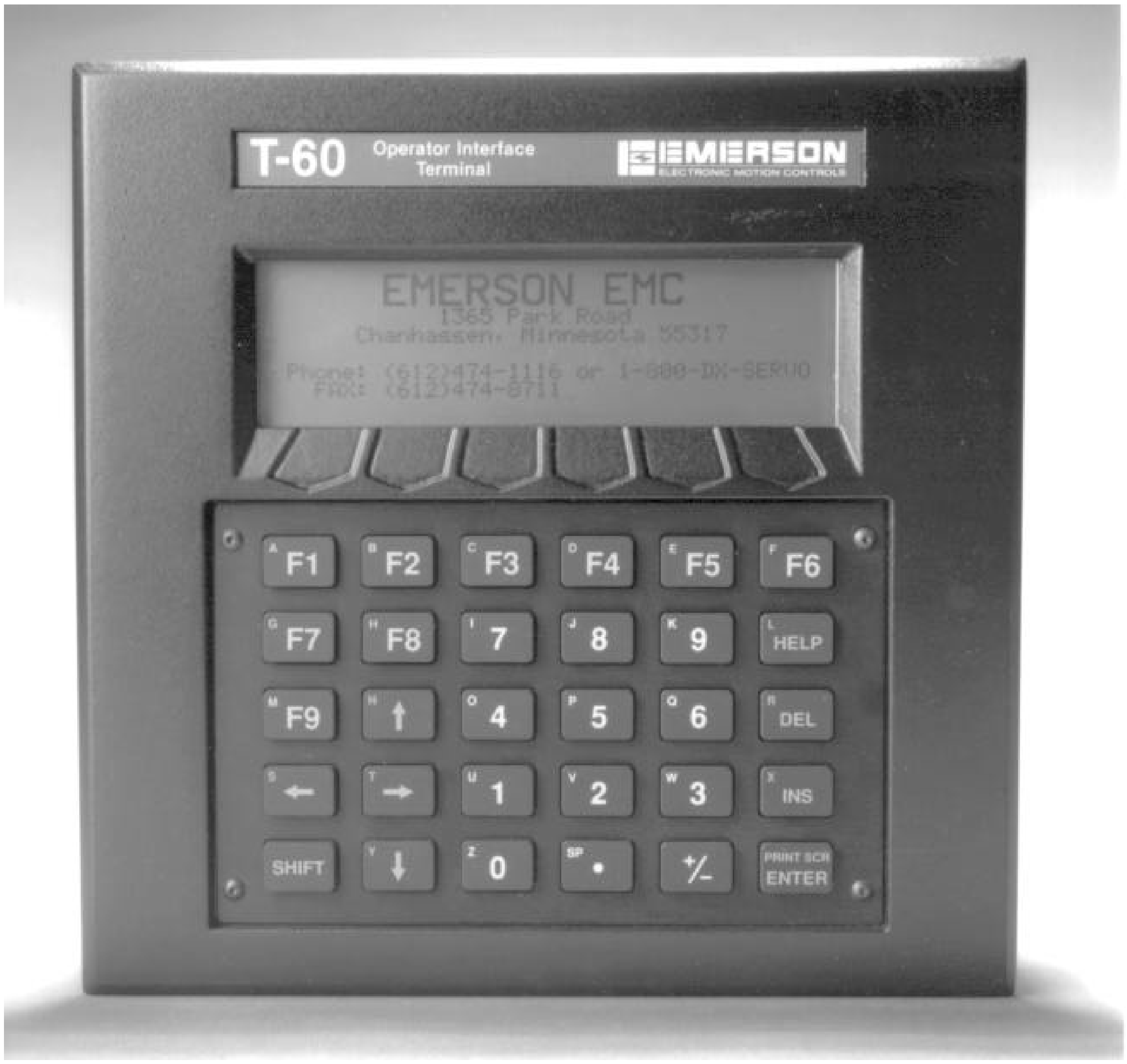
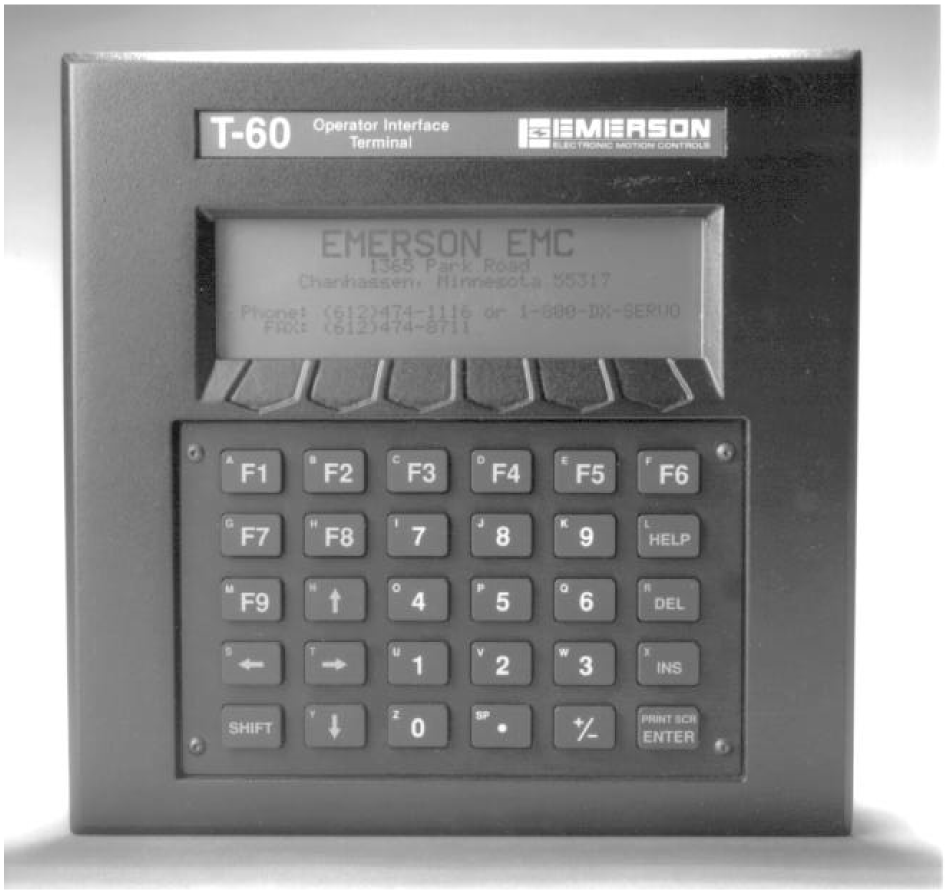
T-60 Operator's Manual


T-60
Operator Interface Terminal
EMERSON
ELECTRONIC MOTION CONTROLS
EMERSON EMC
1365 Park Road
Chanhassen, Minnesota 55317
Phone: (612)474-1116 or 1-800-DX-SERVO
FAX: (612)474-8711
F1
F2
F3
F4
F5
F6
F7
F8
7
8
9
HELP
F9
4
5
6
DEL
1
2
3
INS
SHIFT
0
•
+/-
PRINT SCR
ENTER

P/N 400248-00 Rev.: A3
Date: November1, 1995

T-60 Operator's Manual


T-60
Operator Interface Terminal
EMERSON
ELECTRONIC MOTION CONTROLS
EMERSON EMC
1365 Park Road
Chanhassen, Minnesota 55317
Phone: (612)474-1116 or 1-800-DX-SERVO
FAX: (612)474-8711
F1
F2
F3
F4
F5
F6
F7
F8
7
8
9
HELP
F9
4
5
6
DEL
1
2
3
INS
SHIFT
0
•
+/-
PRINT SCR
ENTER

P/N 400248-00

Rev.: A3
Date: November 1, 1995

Customer Services

EMERSON EMC offers a wide range of services to support our customer’s needs. Listed below are some examples of these services.

Service Support (612) 474-8833

Emerson Electronic Motion Control’s products are backed by a team of professionals who will service your installation wherever it may be. Our customer service center in Minneapolis, Minnesota is ready to help you solve those occasional problems over the telephone. Our customer service center is available 24 hours a day for emergency service to help speed any problem solving. Also, all hardware replacement parts, should they ever be needed, are available through our customer service organization. Need on-site help? EMERSON EMC provides on-site service, in most cases, the next day. Just call EMERSON EMC’s customer service center when on-site service or maintenance is required.

Training Services (612) 474-1116

EMERSON EMC maintains a highly trained staff of instructors to familiarize customers with EMERSON EMC’s products and their applications. A number of courses are offered, many of which can be taught in your plant upon request.

Application Engineering (612) 474-1116

An experienced staff of factory application engineers provide complete customer support for tough or complex applications. Our engineers offer you a broad base of experience and knowledge of electronic motion control applications.

Bulletin Board System (612) 474-8835

EMERSON EMC maintains a BBS which provides you access to software updates, and technical information and services.

Communications protocol:	300-14,400 baud, N, 8, 1

FAX (612) 474-8711

Table of Contents

Introduction

Features

- 8 line by 40 character backlit LCD display with graphics.
- Powerful I/O capability.
  - 2 serial ports (1 port can be RS232, RS422, OR RS485).
  - 8 parallel I/O lines.
  - IBM keyboard interface.
- All I/O is optically isolated.
- Fully featured BASIC programming language.
  - Interrupt capability.
  - Floating point math.
  - Formatted data entry graphics.
  - Easy to use commands for on-board I/O.
- Programmable, context sensitive help key.
- NEMA4 panel mount or wall mount housing.
- Powerful PC-based ApplicationBuilder software included.
- Automatically builds BASIC programs.
  - Place text on the T-60's screen in a wordprocessor like manner.
  - Build comples programs without in-depth knowledge of BASIC.
  - Exchange programs between the PC and the T-60.
  - Built-in terminal emulator.
- Large full-travel 30 key waterproof keypad.
  - 9 soft keys.
  - Tactile feedback.

T-60 Overview

The T-60 Operator Interface Terminal allows you to set up and operate EMERSON EMC positioning servo drive products. The T-60 provides overall control and operator interface for any type of controller or computer which needs an easy to use, intelligent operator interface.

With a T-60 an operator can view and change machine parameters or follow instructions to perform operations. Operators do not have to set switches, thumbwheels or indicator lights. A back-lit 8 line by 40 character "super-twist" LCD display and a large full-travel 30 key waterproof keypad prompts and "listens" to the operator through machine operations. By programmming the Help functionkey, operators have as little or as much "HELP" information as required. The T-60's LCD screen displays TEXT or GRAPHICS providing the operator with block diagrams, flow charts, wiring diagrams and statistical information.

For machine control, the T-60 provides 8 lines of parallel I/O, two serial ports (one is software configurable to be either RS-232 RS-422, or RS-485), and 8 timers to facilitate machine monitoring and control functions. All I/O's are optically isolated and designed to be extremely noise tolerant.

Included with the T-60 is a disk containing the ApplicationBuilder; a PC DOS program which allows you to quickly generate programs for the T-60. The ApplicationBuilder generates BASIC programs from simple menu selections and direct screen entry of text. It consists of three components:

- BUILDER - converts menu selections and direct screen text entry into BASIC code
- EDITOR - performs text editing of BASIC or other ASCII files
- TERMINAL EMULATOR - performs "dumb" terminal operation to talk to the T-60.

Integrated into all ApplicationBuilder functions is an UPLOAD/DOWNLOAD capability and a comprehensive, context sensitive HELP system.

Functional Description

The T-60 is housed in a rugged cast housing which can be flush mounted to an equipment panel. A full gasket and a rigid mounting system forms a water tight seal about the opening. The display is sealed and the keypad is constructed of a water tight silicone rubber. If being water tight is not critical to your application, the T-60 can be wall mounted with supplied brackets.

The keypad on the front of the T-60 is organized into three color coded groups:

WHITE	Numeric Entry
BLUE	Action - CURSOR, ENTER, INSERT, DELETE, HELP
YELLOW	Function Keys

The 8 line by 40 character LCD display serves as a display port, programming tool and soft key label. In the edit mode, the display can be used to scroll through text, make changes, or debug programs. In the run mode text, soft key titles and graphics can be displayed under program control.

The bottom of the T-60 incorporates the entire connector system for parallel and serial I/O.

Figure 1
Block Diagram

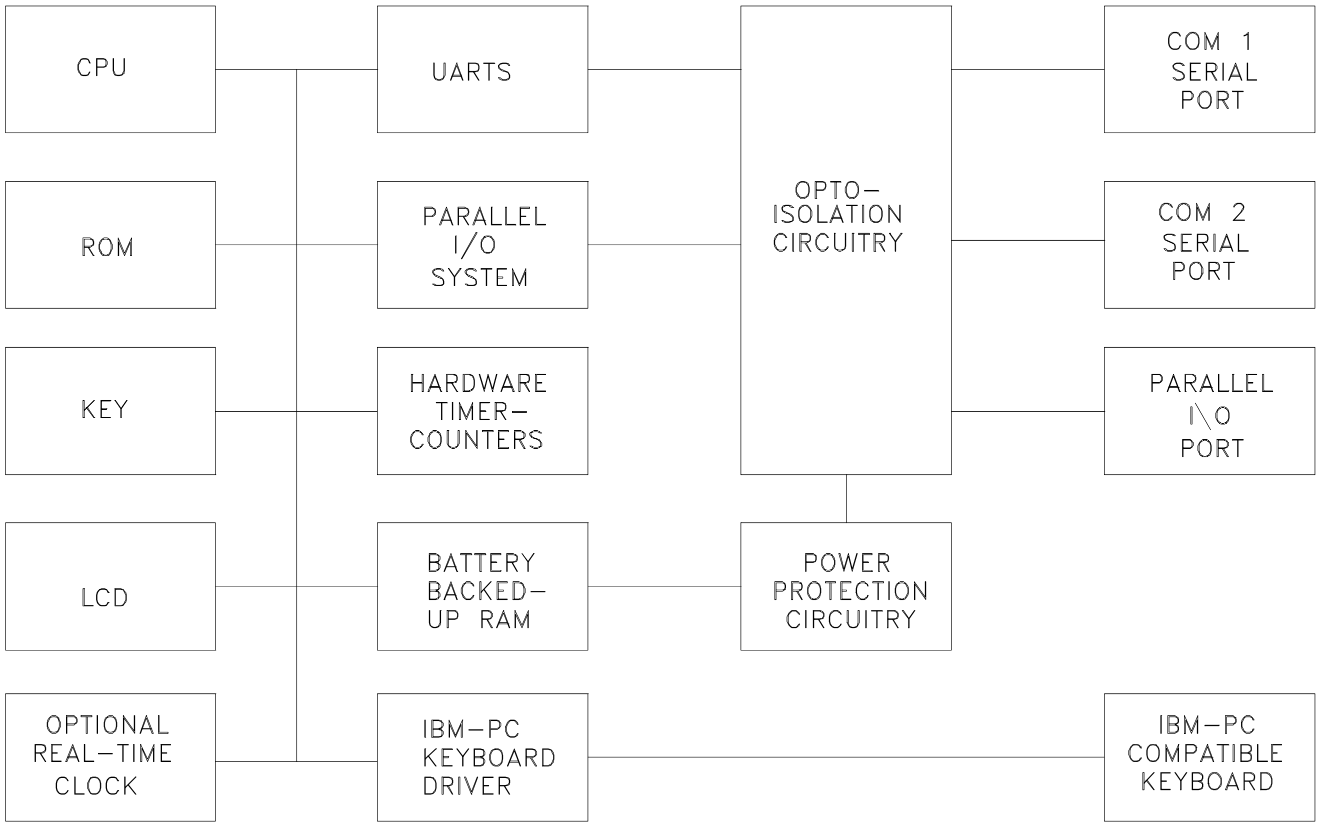


Figure 1 diagrams the internal components of the T-60. The heart of the unit is a high speed 64180, 8 bit high integration CPU chip. The CPU communicates with UARTS (serial ports), ROM, RAM, TIMERS, and I/O. Battery backed-up RAM stores programs and variables even if the power is removed. It can hold a program for over 5 years without power applied. The opto-isolation circuitry is designed to provide a barrier between the outside world (I/O ports) and the CPU. This eliminates CPU errors in high noise environments. The serial and parallel ports have their own power supply to further insure that noise does not disrupt the CPU's operation.

Installation

This chapter focuses on the steps necessary to unpack and install the T-60. Read this section before attempting to apply the T-60. System installers should read this chapter before attempting to install the unit into a cabinet, or before connecting any electrical power to the T-60.

Unpacking and Inspection

Inspect the T-60's shipping container. Is there evidence of damage or mishandling? If damage exists contact your shipping carrier immediately. EMERSON EMC cannot be held responsible for damage in shipment. Compare the contents of the container with the packing list which is attached to the exterior of the shipping container. Your T-60 shipping container should include the following:

- T-60 with installed options
- This manual
- EMERSON BASIC PROGRAMMING GUIDE
- ApplicationBuilder diskettes
- Mounting clips (four)
- Mounting brackets (two)
- 1/4" 8-32 Screws (four)
- Mounting template
- Optional cables and connectors as indicated on the packing list.
- A null modem cable
  - NMA 9 pin to 9 pin
  - NMX 25 pin to 25 pin
- TIA-XXX T-60 to DX Amplifier cable

If any items are missing or damaged, contact EMERSON EMC immediately.

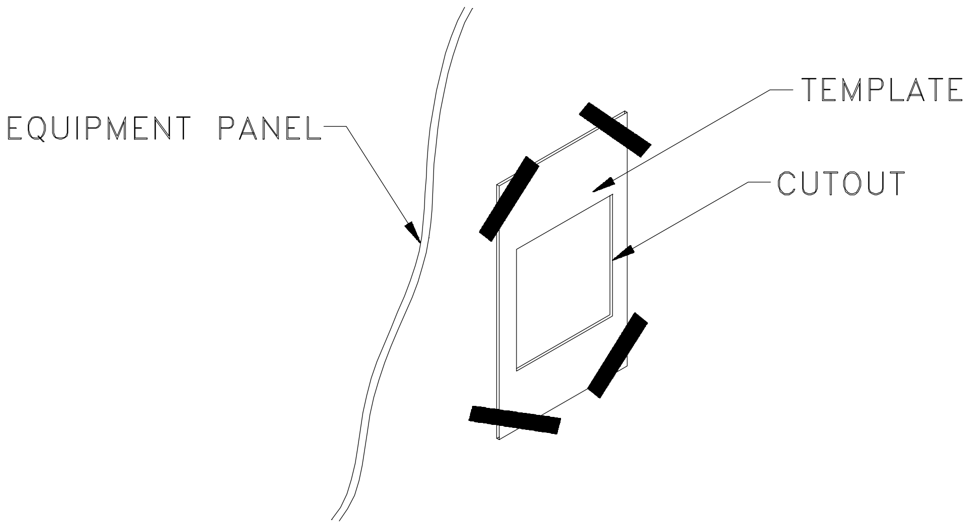
Through Panel Mounting

The T-60 is designed to be mounted either through an equipment panel (in a panel cut-out) or on a flat surface. The through-panel mounting will allow the T-60 to meet NEMA4 specifications for water resistance, and will also resist dust, dirt and non corrosive chemicals. Improper installation could result in damage to the T-60 and other equipment installed in or adjacent to the panel containing the T-60. For safety reasons please follow these instructions closely.

1. Prepare the opening in the panel.
   a. Tape the enclosed mounting template to the front of the panel in the desired location (see Figure 2).
   b. Drill 3/8 " inside the cutout to facilitate cutting.
   c. Use a sabre saw or some other type of sheet metal cutting device to cut out along the "cut here" line.
   d. Using a file, carefully remove any burrs or rough edges that may cut or scratch during the remainder of the installation.
   e. Remove the paper template and discard.

2. Carefully insert the T-60 into the hole in the panel from the front side (see Figure 3).

Figure 2
Panel Cut-Out



3. Hold the T-60 to the panel, and insert the mounting clips as shown in Figure 4.
   a. If the T-60 is to be used in continuously wet applications, we recommend that the installer apply a silicone sealer to the gasket prior to installation.

4. Tighten the mounting clips to secure the T-60 to the front panel.

5. Inspect that the T-60 fits snugly up against the front panel, and that there are no gaps or holes that may allow water or dirt to enter the cabinet.

Figure 3
Mounting

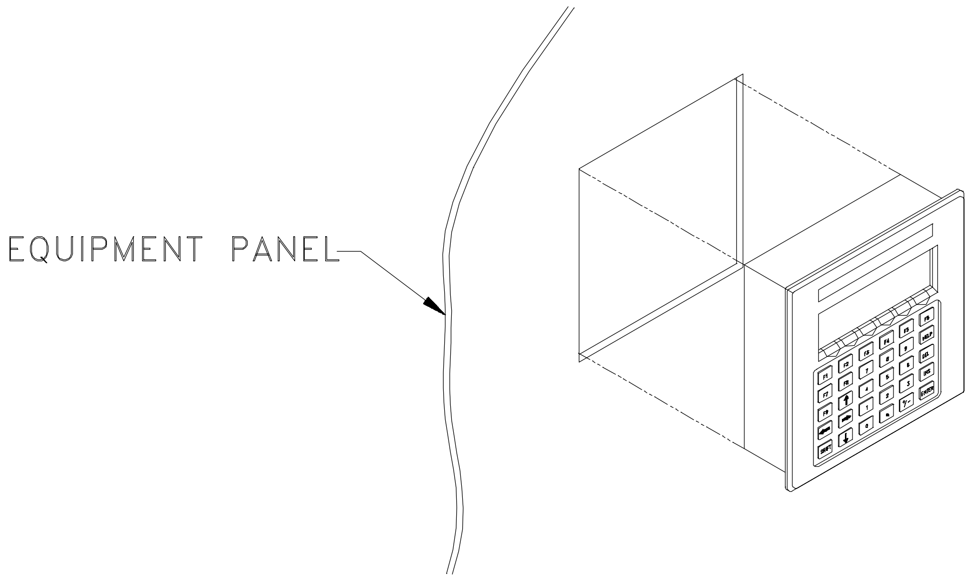
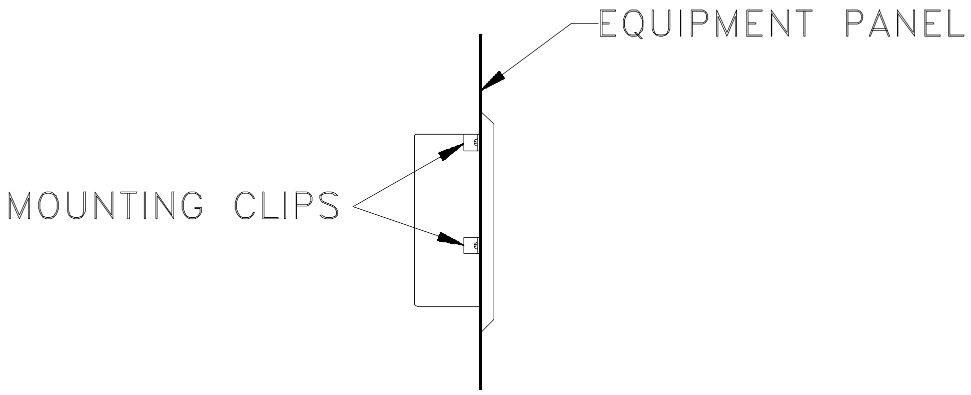


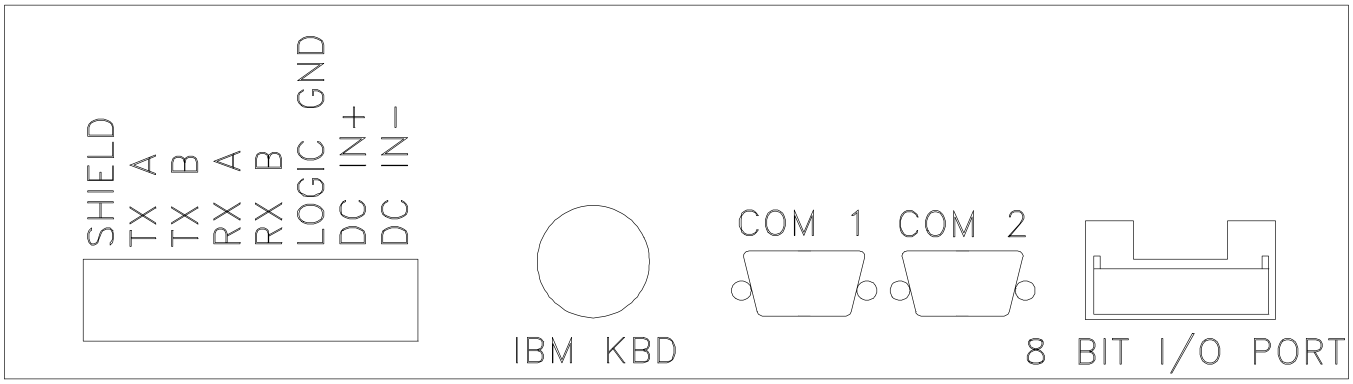
Figure 4
Mounting Clips



System Interconnect

The T-60 is designed to meet a variety of applications; therefore, it is impossible to describe a “standard” interconnect for the T-60. This section will describe each individual connector on the T-60 and how to apply them.

Figure 5
T-60 Bottom Panel



Power / RS422 Connector

The Power/RS422 Connector is an 8 pin screw terminal type connector which provides the DC input power and the RS422 / RS485 connections. The T-60 can accept from 9V to 30V DC at 6 watts. The T-60 actually tries to begin working as soon as the power source passes 4.5 V. This means that at start-up the current requirements from the DC power source can exceed 1.3 Amps. This current surge may cause some power sources to enter current limit rather than to begin working. For this reason it is recommended that the T-60 be used with a power source which is uneffected by this type of current surge. Unregulated supplies, linear power supplies, or high current (2 Amp) switching supplies are recommended.Only connect the DC power source to the DC IN + and the DC IN -. These inputs are reverse protected. DO NOT connect the DC power ground to LOGIC GROUND. Logic Ground, as with all I/O is optically isolated from the I/O. Connecting Logic Ground to DC IN - will result in lower noise immunity.

For connections to the RS422 connector, refer to section “RS422 and RS485 Connections”.

IBM PC-XT Keyboard Connector

An industry standard IBM PC XT keyboard can be connected into the 5 pin DIN connector on the bottom panel of the T-60. Some keyboards have switches which select XT or AT mode. Be sure that the switch is in the XT mode. Some “auto-switch” type keyboards may not work with the T-60. If the keyboard you select is an universal keyboard, make sure it has a switch.

Note that the IBM KEYBOARD connector is not optically isolated, and use in a high noise environment should be avoided. The primary use for this connector and the IBM keyboard is for program development. If necessary, remove the keyboard from the connector while running your application.

COM1 and COM2 Connectors

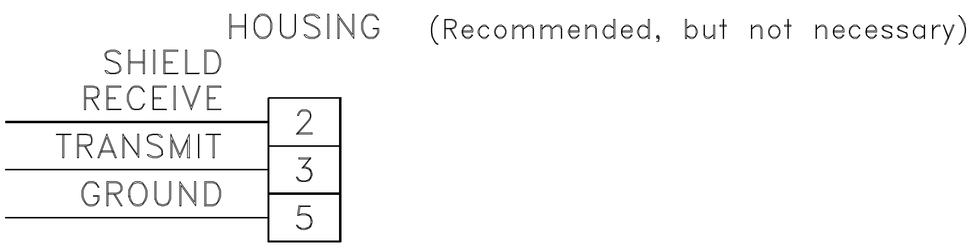
COM1 and COM2 are the serial port connectors (DB-9 style, male). The pin-outs for these connectors are as follows:

Table 1
COM1 and COM2 Connectors Pin-Outs

Pin:	**Connection:**
Housing	SHIELD
2	RS232 RECIEVE (INPUT)
3	RS232TRANSMIT (OUTPUT)
5	SIGNAL GROUND

These connectors provide connection for serial communications between the T-60 and a variety of devices. Connections made with this connector are most commonly referred to as SERIAL PORTS. Examine your application carefully, and make sure that you have TRANSMIT on the T-60 connected to the RECEIVE of the device under control, and RECEIVE of the T-60 connected to the TRANSMIT of the device under control.

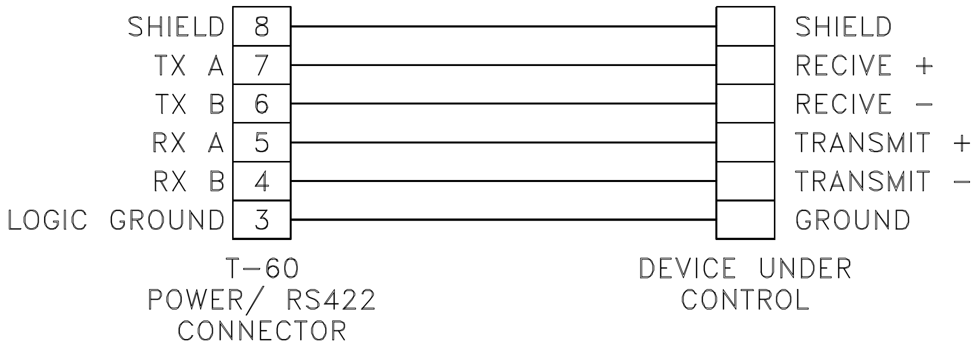
Figure 6
RS232 Minimum Connections



RS422 and RS485 Communications

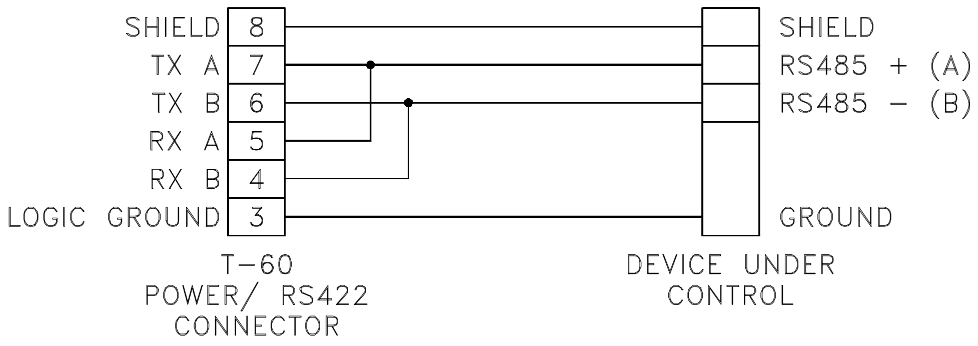
The T-60 can communicate via RS422 or RS485 on COM1. When either of these modes are selected, the RS232 COM1 connector is disabled. To use the RS422 connections, follow the same conventions outlined for RS232, connecting outputs on the T-60 to inputs on the device under control, and inputs on the T-60 to outputs on the device under control. To enable the RS422 transmitter, refer to the RS422 Statement in the EMERSON EMC Basic Programming Guide. Figure 7 depicts proper RS422 connections.

Figure 7
RS422 Connections



To perform RS485 serial communications with the T-60, the RS422 transmitter and receiver lines must be tied together. Since the RS422 transmitters are disabled after power up, it is safe to assume that no line driver conflicts will occur. Again, refer to the RS422 Statement in the EMERSON EMC BASIC PROGRAMMING GUIDE. The electrical connection between the T-60 and the device under control are shown in Figure 8.

Figure 8
RS485 Connector



I/O Connector

The I/O connector on the T-60 is designed to provide 8 bits of parallel I/O. Each line can be configured in software to perform input, output, or bidirectional operations. The pin-out is an industry standard 20 pin ribbon cable, compatible with industry standard PB-8 boards. Pinout as follows (even numbered pins - logic ground):

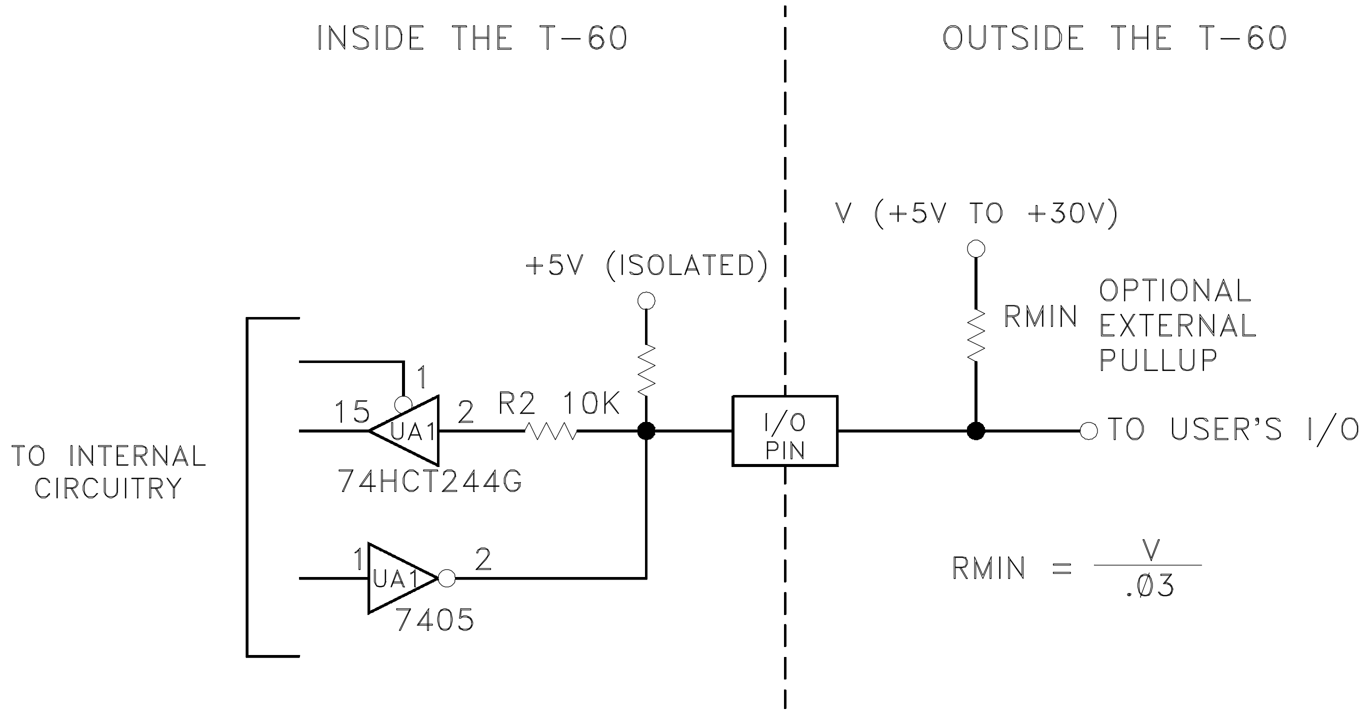
Table 2
I/O Connector Pin-Out

Pin:	**Function:**
19	+5V DC OUT*
17	I/O 0
15	I/O 1
13	I/O 2
11	I/O 3
9	I/O 4
8	I/O 5
5	I/O 6
3	I/O 7

The schematic in Figure 9 represents one I/O bit on the 8-bit I/O interface. All of the other 7 bits are identical. U1 is a CMOS receiver which is constantly monitoring the status of the I/O line. Its input is current limited by R2 and pulled up by R1. A current limit (R2) is needed to allow the I/O pin to exceed the supply voltage of U1 without damage. Notice that the pull-up (R1) is a fairly high value (22K ohms). This allows the user to connect any voltage from 5 to 30VDC to the pin without damage to the pull-up resistor. U1 senses a logic high (1) on the I/O pin at 3.2 V and a logic low (0) on the I/O pin at 1.4 V regardless of the externally applied pull-up voltage.

U2 is an open collector output driver which can withstand 30 V and 30 MA load. When a BASIC program writes a 1 to the output port, U2 is off, allowing the pull-up resistor (R1) or the user's externally connected pull-up to pull the I/O line high. It is possible in this condition for an external device like a switch to pull the line low thereby allowing the input U1 to sense a logic low. When a BASIC program writes a 1 to the output port, the output transistor in U2 turns on, thereby driving the I/O line low. When the T-60 powers up, all I/O lines are driven to a 1, thereby allowing the I/O pins to operate as inputs.Use the OUT and INP commands to program the T-60's 8 bit I/O port. They are described in detail in the EMERSON EMC BASIC PROGRAMMING GUIDE.

Figure 9
One Channel of the 8 Bit I/O



Support for the T-60's 8 bit I/O has been provided by the following commands. Some commands refer to 24 bits of I/O instead of 8. This reflects the compatability between the T-60 and the T-61 (the T-61 has 24 bits of I/O standard while the T-60 has 8 bits of I/O). The T-60 can only use the lower 8 bits of these commands.

INP	Input port data
IO24	Specify an I/O bit pattern to generate an interrupt.
ON IO24	Interrupt on an I/O bit pattern
OUT	Output port data

These commands are described in the EMERSON EMC BASIC USERS GUIDE in a manner which works with both the EMERSON EMC T-61 and T-60. The specific changes are as follows:

INP	Input range of 0 to 255.
IO24	Bit pattern ranges: “0”, “1” or “X” through “00000000”, “11111111”, or “XXXXXXXX”. the input may still be a string variable.
ON IO24	Unchanged.
OUT	Output range 2 to 255. The MAP function works over the range of: “0”, “1” or “X” through “00000000”, “11111111”, or “XXXXXXXX”.

There are similar restrictions on the use of the EVENT DRIVEN SOFTWARE. I/O ranges are limited to 8 bits. If you have questions about the EVENT DRIVEN SOFTWARE and the T-60, please contact EMERSON EMC.

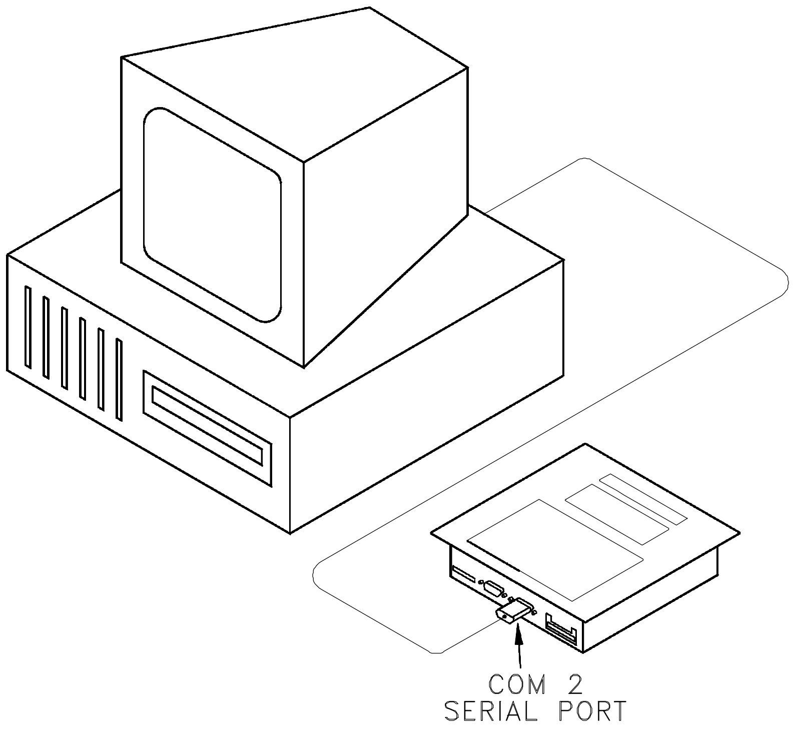
ApplicationBuilder

Included with your T-60 is a disk containing the ApplicationBuilder, a PC program which allows you to quickly generate programs for the T-60. The ApplicationBuilder generates BASIC programs from simple menu selections and direct screen entry of text. It consists of three components: the BUILDER, which converts menu selections and direct screen text entry into BASIC code; the EDITOR which performs text editing of BASIC or other ASCII files; and a TERMINAL EMULATOR which can perform "dumb" terminal operation to talk to the T-60. Integrated into all ApplicationBuilder functions is an UPLOAD/DOWNLOAD capability and a comprehensive, context sensitive HELP system.

Getting Started

Running the ApplicationBuilder is quite simple, and requires no special software skills. First you must connect the T-60 to your PC. Connect one end of the NULL MODEM cable to COM2 on the T-60 and the other end to an available serial port on your PC (either COM1 or COM2). Note which serial port on the PC that you have connected to (for information on the serial ports for your PC refer to your PC's instruction manual).

Figure 10
Serial Connections



The ApplicationBuilder is supplied on two 5 1/4" diskettes and on one 3 1/2" diskette. Make back-up copies of your diskettes before attempting to use them. Save your original ApplicationBuilder diskettes in a safe place.

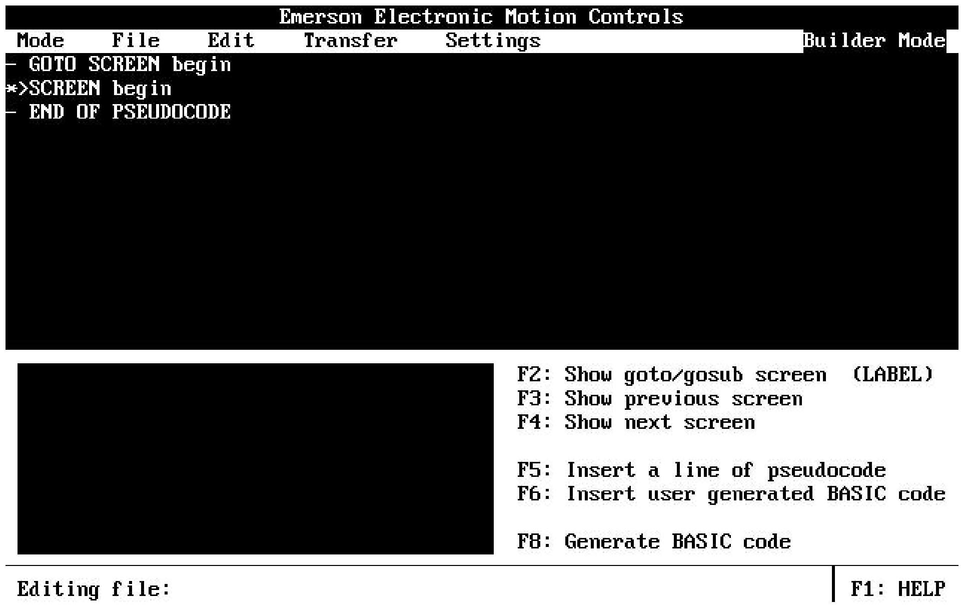
If your system has a hard drive, copy the contents of either of the diskettes onto the hard drive (make a separate directory first for ease of use). If you do not have a hard drive, place the disk labeled DISK 1 into your diskette drive.Log onto the drive with the ApplicationBuilder on it (i.e.. type A: or B: or C: followed by Enter). Type BUILD followed by Enter to run the ApplicationBuilder. If you wait a few minutes and your screen remains blank, remove the disk and reboot your computer. Re-run the ApplicationBuilder by logging onto the dirve containing the ApplicationBuilder and typing BUILD X followed by Enter.

The ApplicationBuilder will start up for the first time in LCD mode (the simplest display mode). If your computer has a color monitor, selecting the SETTINGS menu by pressing ALT-S, and choose the COLOR selection by moving the cursor to the COLOR text followed by ENTER. This will display ApplicationBuilder screens in color. The change you have just made will be recorded to a configuration file on disk, so subsequent operation of the program will be in color.

To select the COM port on your PC which is connected to the T-60, press ALT-S to pull down the SETTINGS menu. Choose either COM 1 or COM 2 depending upon which port the null modem cable is connected to on your PC. Perform the selection in the same manner described above. Again, the settings will automatically be saved for you when you exit the ApplicationBuilder program.

Figure 15 shows the layout of the BUILD screen. It is comprised of four basic components. At the top there is the MENU BAR. This is where the pull-down menus for operating the ApplicationBuilder originate. In the center is the PSEUDOCODE GENERATION area. This is where English language program statements which we call PSEUDOCODE are placed and edited. In the lower left is the SIMULATION of the T-60's screen. This represents the 8 line by 40 character LCD screen on the T-60. The lower right contains descriptions for the function keys. These keys are used to generate the PSEUDOCODE.

Figure 11
Builder Mode Screen



```
- GOTO SCREEN begin
*>SCREEN begin
- END OF PSEUDOCODE
```

```
F2: Show goto/gosub screen  (LABEL)
F3: Show previous screen
F4: Show next screen

F5: Insert a line of pseudocode
F6: Insert user generated BASIC code

F8: Generate BASIC code
```

```
Editing file:                                     F1: HELP
```

Simple Example

To demonstrate the power of the ApplicationBuilder, a very simple example program will be read from disk, viewed, converted to BASIC, downloaded, and run.

Select the FILE menu (ALT-F) and OPEN (move the cursor or type O) followed by ENTER. You will see a window appear which contains a listing of the demonstration builder files on the ApplicationBuilder diskette. Move the highlight to EASY.BLD and press ENTER. The PSEUDOCODE for EASY.BLD will appear on the screen. Scroll up and down through the PSEUDOCODE with the cursor keys. Here is a listing of the PSEUDOCODE for EASY.BLD:

```
- GOTO SCREEN begin
*>SCREEN begin
- PUT TEXT AT (7,7): “PRESS THE ‘MORE’ SOFTKEY”
- PUT LARGE TEXT AT (3,3): “TO THE BUILDER”
- PUT HUGE TEXT AT (2,1): “WELCOME”
- SOFTKEY (1) “MORE” GOTO SCREEN morescr
- SOFTKEY WAIT
*>SCREEN morescr
- PUT LARGE TEXT AT (3,1): “EMERSON EMC”
- PUT LARGE TEXT AT (2,2): “MAKES  PROGRAMMING”
- PUT HUGE TEXT AT (3,2): “EASY!!”
- DELAY 2000
- GOTO SCREEN begin
- END OF PSEUDOCODE
```

Observe the first two lines and the last line of the PSEUDOCODE. These three lines are automatically inserted by the BUILDER. You may have noticed them on the screen before you retrieved EASY.BLD from disk. These lines cannot be deleted, edited or moved. They are essential for the builder’s operation. Notice that each line begins with a dash (-) or a * sequence. These tell the BUILDER that the line is PSEUDOCODE, and what to do with it.

A line starting with * is a new screen, while a line starting with a dash is PSEUDOCODE that will execute for that screen. Don’t worry about generating these special symbols, the BUILDER does that for you.

Next move the cursor (blinking block) over the “*SCREEN begin” PSEUDOCODE. Notice that the SIMULATION OF THE T-60’S SCREEN contains text. This is the text which will be placed on the T-60 actual screen when BASIC is generated and the program is run on the T-60. Now try placing the cursor over the fourth line:

```
- PUT TEXT AT (7,7): “PRESS THE ‘MORE’ SOFTKEY”
```

Press ENTER. Notice that the “PRESS THE ‘MORE’ SOFTKEY” is highlighted on the T-60’s screen. If you hold down the SHIFT key and move the cursor key, the text block will move on the screen. Notice that the two numbers in parentheses are changing. This action changes the place where the BUILDER will place the BASIC text string when the PSEUDOCODE is converted to BASIC.Press ESC (leave the program unchanged). If you were to press ENTER, new position would have been saved.

Press SOFTKEY F5. This is how you select PSEUDOCODES to place on screen. Type an S. Press ENTER. The SCREEN PSEUDOCODE has been selected. This allows you to type text directly on the screen. Press ESC (leave the program unchanged).

Press SOFTKEY F8. This softkey generates BASIC. You may be asked if you wish to write over the existing BASIC program: answer “Y” for YES or press ENTER (if the ApplicationBuilder has been used before, the file EASY.BAS may already exist on the disk). Notice that the number of basic lines generated are displayed on screen during this process. You will be asked if you want to load the BASIC program into the EDITOR, answer YES. You will be asked if you want to save EASY.BLD, answer NO since you did not change anything (or mean to change anything!).

The BASIC program generated by EASY.BLD is now loaded into the EDITOR. The EDITOR has the normal attributes of a text editor. CUT / PASTE / INSERT / etc. To select a block of text to cut or copy, move the cursor to the start of the text block and hold down the shift key and move the cursor to the end of the block. The selected block will be highlighted. You may now CUT copy COPY the text in the selected block by using the functions in the EDIT menu. Try pressing F1 to further examine the capabilities of the EDITOR. After examining the BASIC code generated, let’s download it into the T-60.

Press ALT-T and select Download Disk File (by selecting it and pressing Enter or by typing “D”). Select EASY.BAS and press ENTER. This begins the download process from the PC to the T-60. While data is transferred between the PC and the T-60, a counter will show the number of bytes being transferred. When the process is complete, the program has been downloaded. Press F1 (RUN) on the T-60. You should see the screens and softkeys that were present on the SIMULATION OF THE T-60’S SCREEN on the PC.

Once you have successfully downloaded and run EASY.BLD, try loading, examining, and compiling other programs on the ApplicationBuilder diskette. Some programs are only available in BASIC form. They may be loaded with the ApplicationBuilder even though their corresponding BUILDER file is not present.

Trouble-Shooting Communications

If your attempt at down-loading a program to the T-60 was unsuccessful, check the serial port connections between the T-60 and the PC. Look for the following things:

1. Is power applied to the T-60?
2. Is one end of the NULL MODEM cable connected to COM 2 of the T-60? If it is not, reconnect it to COM 2.

3. Which port on the PC is the other end of the serial cable connected to: COM 1 or COM 2? Does this match with the selections made in the SETTINGS menu? If you are unsure, recheck the SETTINGS menu (ALT-S) and make sure that you have the correct setting.

4. Are the NULL MODEM cable connectors securely seated into their respective sockets? Try tightening the hold-down screws on the NULL MODEM cable.

5. Are you using the NULL MODEM cable supplied by EMERSON EMC? If not, check the cable for correct configuration.

6. Do you have memory resident programs loaded (TSR programs) which are presently running on your PC (disk spoolers, communications drivers, etc.)? If so disable or remove them. TSR's may cause problems with the serial port operation.

Operating Modes

This portion of the manual has been created from excerpts of the on-line HELP which is integrated into the ApplicationBuilder program. If you feel comfortable with the discussion so far, go ahead and try the ApplicationBuilder on your own. If you need help at any time, press F1. A help message will appear which will describe the operation required to run the ApplicationBuilder. After trying out the ApplicationBuilder, read this section of the manual to master its power. Section "Summary of ApplicationBuilder" contains a keystroke summary.

The ApplicationBuilder consists of three basic operating modes: BUILDER, EDITOR, and the TERMINAL modes. These modes are selected by the MODE menu. When the ApplicationBuilder begins, it starts up in the BUILDER mode. To change modes, use the MODE menu. The Mode menu gives you access to the Builder, the Editor, and the Terminal Modes of the ApplicationBuilder. You can switch from mode to mode without closing the current document allowing you to work on a number of things at once. You also quit the program from the Mode menu. You can access the Mode menu by hitting the <Esc> key or Alt-M.

BUILDER - The Builder Mode is the mode you utilize to use a user friendly program generation environment to help you develop your BASIC program to run on your T-60. Typing "B" while anywhere in the Mode menu will also activate this command.

EDITOR - The Editor Mode is basically a standard text editor where you can develop and modify your BASIC programs. Not only can you cut, copy, and paste blocks of text, but you can also renumber your BASIC programs in the Editor. Typing "E" while anywhere in the Mode menu will also activate this mode.

TERMINAL - The Terminal Mode basically turns your computer into a dumb terminal that communicates directly to the T-60's interface. For all practical purposes, key strokes entered onto your computer keyboard are echoed on the interface screen. Typing "T" while anywhere in the Mode menu will also activate this command.

RUN TUTORIAL - This selection provides a guided tour of the ApplicationBuilder. Text will be placed on-screen with examples of how to run the ApplicationBuilder.

QUIT - Exits the ApplicationBuilder program and returns you to DOS. Typing "Q" while anywhere in the Mode menu will also activate this command.

Builder Mode

The Builder Mode is the heart of the ApplicationBuilder package. It allows you to easily develop a control and interface program, test the operator interface functionality of the program, and then generate the proper BASIC code to run in a T-60. You accomplish this by generating “Pseudocode”. You can enter a line of Pseudocode by pressing F5, finding the appropriate Pseudocode for the task you want to accomplish, pressing Enter, and following the instructions for entering the requested data. You write a program in Pseudocode that represents what you want the interface and control program to do. Not all of the functionality of the Emerson BASIC is represented in Pseudocodes. At any time in the process of generating the program you find that you need functionality that the current Pseudocodes don’t provide, you can press F6 and enter BASIC directly. Once you have a Pseudocode (and possibly Basic) representation of what you want your program to do, you press F8 which converts your Pseudocode program into a line numbered BASIC program.

The EDITOR functions are available while operating the BUILDER. You can cut and copy text blocks from one section of your Pseudocode to another. Refer to the EDITOR for operation examples.

Operating the Builder

The box in the lower left hand corner of your computer screen is a representation of your T-60’s screen. While you are generating your Pseudocode program, any line of code that generates information that will be displayed on the LCD display will show up here. Text, soft key labels, placeholders for variable and operator numeric entry will all show up and can be moved to any legal location you desire. You will find that as you move the cursor down from Pseudocode line to Pseudocode line, the screen will build up one line at a time in synchronization with the Pseudocode line you are on.

There are a few other important keys and keystrokes to keep in mind. Ctrl-Y will delete the line of Pseudocode where the cursor is currently located. Cut, Copy and Paste are all functions which allow blocks of Pseudocode to be manipulated. Pressing F5 when the cursor is on a line of Pseudocode will add a new line of Pseudocode above the cursor location. Pressing Enter while on a line of Pseudocode will allow you to edit that line.The Pseudocode you generate will be organized into Screens. Each screen will have a name (defined by the SCREEN Pseudocode). The GOTO screen and SOFTKEY Pseudocodes will allow your program to move from screen to screen. F2, F3, and F4 help you navigate through your Pseudocode program as you are developing it. If you are on a line of Pseudocode that defines a GOTO or GOSUB to a screen or a label and press F2, the cursor will move to the beginning of that screen or to that label. F4 will send you to the beginning of the next screen definition. F3 will send you to the beginning of the previous screen definition. These function keys become increasingly helpful as your programs get longer and more involved.

Entering Pseudocode falls into two categories: Entering/placing text on the screen and everything else. Pseudocodes that have nothing to do with text on screen are relatively simple. Each bit of information needed to complete a Pseudocode command is asked for one piece at a time. You can get from entry to entry with the Tab key or the Enter key. The Enter key causes you to exit the Pseudocode entry box once everything has been defined. The Tab key takes you from entry to entry and from the last entry to the first without exiting the screen. Shift-Tab takes you backwards.On Screen entry allows you to do even more powerful things. Any command that generates text to be placed on the screen takes advantage of On Screen Editing. The SCREEN command allows you to place regular text, large text and huge text at any location on the screen. When entering text on screen, you are always in overtype mode. You change text sizes with the Tab key. Once you are done typing the text to be displayed on the screen, you press Enter and are then prompted for a name for the screen. Once that is done (and you press Enter again) you will see that a series of PUT TEXT commands have been generated. Whenever you are entering a new PUT TEXT command or editing an existing one, your cursor will be on the screen. You can add or change text or move it around. To move text, you hold down the Shift key and then press Cursor keys to move the text to the desired location. Once you have the proper text on the screen in the proper location, press Enter to generate the Pseudocode.

Certain Pseudocodes (like PUT NUMBER) allow you to specify numeric formats for printing numbers on the screen. You may specify a digit place holder with a # character. Use the decimal point (.) character for specifying the location of the decimal point. Use a plus(+) or minus (-) sign to specify the use of a sign. If you want scientific notation, use an E. For example:

##.###	Specifies from 0.000 to 99.999
+#.#	Specifies from -9.9 to +9.9
###	Specifies from 0 to 999

PUT TEXT has the capability to place text from string variables directly on screen. The actual process to place the variable may seem a little confusing at first. To place the contents of a string variable with the PUT TEXT Pseudocode, press F5 followed by P. Select PUT TEXT with the cursor, press Enter. You are now in “ON SCREEN” text entry (i.e. your cursor is on the simulated T-60 screen and blinking). Move the cursor to the desired starting location of the text string. Type a dollar sign ($) followed by Enter. A new window will appear in the Pseudocode entry area which will ask for the name of the string variable (a string variable must end with a dollar sign ($)). Enter the string variable and press return.

Selecting Pseudocode is simple. Pressing F5 displays a list of Pseudocodes that you can choose. The Pseudocode selection screen displays the Pseudocode list, the currently selected Pseudocode, and some helpful information about the currently selected Pseudocode. If you press Enter, you will go to the entry screen (if one is needed) for the currently selected Pseudocode. You can use the Up and Down Arrows, as well as the Page Up and Page Down keys, to scroll through the Pseudocodes. You can also press a letter key to get to the first command in the Pseudocode list that begins with that letter. For instance, if you are in the list and press S, you will go to the SCREEN command location in the Pseudocode list.

Editor Mode

Editor Mode is a text editor allowing you to edit files off line from any serial device (such as a T-60.) This allows you to write and modify programs away from your machine and to use more powerful editing features than BASIC has built in. With the Editor, you can easily Cut, Copy, and Paste blocks of text from one area or program to another. You can even cut or copy text in the Editor and paste it into the Builder and vice versa.

To select text to cut or copy, you position the Cursor at one end of the block of text in question. Then hold down the Shift key as you move the cursor with the Arrow keys and Page up/Page down keys. You then select Cut or Copy from the Edit menu. You can also use Shift-Delete or Ctrl-Insert, respectively, to perform the same functions. You can Paste the text anywhere you put the cursor (or over any block of text you choose) by choosing Paste from the Edit menu or by using Shift-Insert.

When you cut or copy text from a document, it is saved in the “Clipboard.” You can view the clipboard at any time by choosing View Clipboard from the Edit menu. You will notice that the last block that was cut or copied is highlighted and that many of the previous cut or copied blocks are still in the clipboard. If you would rather paste a previously cut or copied block of text, just highlight that block, return to your document, and paste that previously (and now currently) chosen text. The following sub menus appear when you select the EDITOR MENU:

CUT - Text removes the currently selected text and places it in the clipboard. It can then be pasted into another location or file. Typing “T” while anywhere in the Edit menu will also activate this command.

COPY - Text takes a copy of the currently selected text and places it in the clipboard. It can then be pasted into another location or file. Typing “C” while anywhere in the Edit menu will also activate this command.

PASTE - Text takes the currently selected text from the clipboard and pastes it into the currently selected location in your document. Unless you explicitly select otherwise in the clipboard, the text that is pasted will be the last block of text cut or copied. Typing “P” while anywhere in the Edit menu will also activate this command.

SHOW - Clipboard opens the clipboard so you can view its current contents. You can also select text other than the most recently cut or copied text to paste into a document just by highlighting a different section of text. Typing “S” while anywhere in the Edit menu will also activate this command.

RENUMBER - Lines acts just like the BASIC “RENUM” command if you have a BASIC file with a .BAS file extension loaded into the Editor. This will be handy if you have to insert a large number of lines of code into a BASIC program in the Editor. Typing “R” while anywhere in the Edit menu will also activate this command.

Terminal Mode

The Up and Down cursor keys are not currently functional. If you want to move up or down on the screen, you must do it from the cursor keys on your T-60. If you are editing a line of BASIC, use the EDIT command which will position the cursor on the line you want to change.

Terminal Mode basically turns your computer into a dumb terminal. Virtually all keys that you hit on your keyboard are sent directly over the serial link from your PC to T-60. Also, anything sent out of the Emerson device's serial port connected to your PC will show up on the screen of your PC when in Terminal Mode. In effect, using the Terminal Mode to communicate with the T-60 is quite a bit like typing on a keyboard plugged into the keyboard port on your Emerson T-60 (the exception being the use of <Ctrl> and function keys).

If you enter Terminal Mode and have problems communicating, verify that you have the correct COM port selected in the Settings Menu and that you have a null modem cable between that COM port and COM2 on your Emerson product. Once you have verified this setup, select Start Communications from the Terminal menu to get things going.

File Menu

The File menu is used to manipulate files. You begin a New file, Open a preexisting file, Save a file, save a file under a different name, or Print a file. If you need to get to DOS briefly, the DOS Shell command allows you to go to DOS and get back easily. You can access the File menu by hitting Alt-F. The following selections appear when you select the File menu:

NEW - closes any currently open file and starts a new one. This only affects the currently active mode. Typing "N" while anywhere in the File menu will also activate this command.

OPEN - allows you to close any currently open file and open the file of your choice. You will be prompted to select from a list of acceptable files or you can type in the path and filename directly. Typing "O" while anywhere in the File menu will also activate this command.

SAVE - immediately saves the currently active file to the most recent filename assigned to it. Typing "S" while anywhere in the File menu will also activate this command.

SAVE_AS - saves the currently active file, but it first prompts you to specify a new path and/or file name. Typing "A" while anywhere in the File menu will also activate this command.

PRINT - prints the currently active file to the default printer. Typing "P" while anywhere in the File menu will also activate this command.

DOS SHELL - Selecting DOS Shell will send you back to DOS to take care of some brief task. Typing "exit" will then return you to the ApplicationBuilder. Typing "D" while anywhere in the File menu will also activate this command.

Transfer Menu

The Transfer menu is used to transfer files back and forth between your T-60 and your PC. You can Download from the PC to the Emerson T-60 or Upload from the product to your PC. You can also Verify whether a program in your interface matches one on your PC or not. You can also Start and verify communication between the two devices. You can access the Transfer menu by hitting Alt-T. The following selections appear when you select the TRANSFER MENU:

DOWNLOAD Disk File **-** transfers a file from your PC to the T-60. You will be prompted to select a currently saved BASIC file, or you can enter in the filename of the program directly. Typing “D” while anywhere in the Transfer menu will also activate this command.

UPLOAD Disk File **-** transfers a file from your T-60 to your PC. You will be prompted to select a filename for the uploaded file. You can save it as an existing filename thereby overwriting the existing file with the uploaded file, or you can assign a new name to the uploaded file. Typing “U” while anywhere in the Transfer menu will also activate this command.

VERIFY Disk File **-** compares a selected file on disk in your PC with the file currently loaded in the T-60. This allows you to determine which versions of your BASIC programs exist in the PC and the Emerson T-60. This is useful if, for instance, you are not sure if you have made changes in your program in the Emerson T-60 which are not saved on disk. If you do a Verify Disk File with the latest version you have on disk, and they are different, you know that something has changed in the T-60 since you last downloaded the program. Typing “V” while anywhere in the Transfer menu will also activate this command.

START COMMUNICATIONS - does a number of things that will be useful to you. First, it stops the execution of any program currently running in your T-60. It then enables remote communication and verifies that the communication link is working correctly. Any time you cannot seem to communicate with your Emerson T-60, verify that you have the correct COM port selected in the Settings Menu and that you have a null modem cable between that COM port and COM2 on your Emerson T-60. Once you have verified this setup, select Start Communications to verify the setup. Typing “C” while anywhere in the Transfer menu will also activate this command.

Setting Menu

The Settings menu is used to configure your PC. You can specify the configuration of your PC’s serial ports and which serial port you will be using to communicate with your T-60. You can also configure the ApplicationBuilder to use colors which match the capability of your computer screen. You can access the Settings menu by hitting Alt-S. The following sub menus appear when you select the SETTINGS MENU:

SELECT COM1 - Highlighting COM1 and pressing 07
selects COM1 as the active COM port in your PC for communication with your T-60. Typing “1” while anywhere in the Settings menu will also activate COM1.

SELECT COM2 - Highlighting COM2 and pressing 07 selects COM2 as the active COM port in your PC for communication with your T-60. Typing “2” while anywhere in the Settings menu will also activate COM2.

LCD - Highlighting LCD Mode and pressing 07 configures the ApplicationBuilder to run effectively on a computer with an LCD based screen (common on laptop or notebook PC’s.) The program uses only black and white colors for visibility. Typing “L” while anywhere in the Settings menu will also put the ApplicationBuilder into LCD mode.

MONO - Highlighting Mono/Composite Mode and pressing 07 configures the ApplicationBuilder to run effectively on a computer with a Monochrome monitor The program uses black, white and intense white colors only for visibility. Typing “M” while anywhere in the Settings menu will also put the ApplicationBuilder into Mono/Composite mode.

COLOR - Highlighting Color Mode and pressing 07 configures the ApplicationBuilder to run effectively on a computer with a color monitor The program uses many colors to enhance the readability of the screen. Typing “C” while anywhere in the Settings menu will also put the ApplicationBuilder into Color mode.

CONFIG COM1 - Selecting Configure COM1 allows you to set the communication parameters for COM1 in your PC. The values you can set are: baud rate, number of data bits, number of stop bits, parity, and whether you want local echo on or off. Once you have selected Configure COM1..., you will see a dialog box for setting these parameters with help information located at the bottom of the screen.

CONFIG COM2 - Selecting Configure COM2 allows you to set the communication parameters for COM2 in your PC. The values you can set are: baud rate, number of data bits, number of stop bits, parity, and whether you want local echo on or off. Once you have selected Configure COM2..., you will see a dialog box for setting these parameters with help information located at the bottom of the screen.

Summary of ApplicationBuilder Operation

The following is a summary of the sequence of operations required to make a functional program in the ApplicationBuilder (this example assumes you are starting from the DOS prompt with the ApplicationBuilder diskette or hard disk ready). Your T-60 should be connected to the PC with a NULL MODEM cable.

Operation:	**Keystroke:**	**Description:**
Running the program:	BUILD	At the DOS prompt, run the ApplicationBuilder program.
	ALT-F	File Menu
	Enter	“NEW” - clear out workspace and begin a new Pseudocode program.
Making the first screen:	CURSOR	Move the cursor to the second line in the Pseudocode list (*SCREEN begin).
	Enter	The cursor is now on the screen. You are now building a new screen named “begin”. You may position the cursor, enter characters or press the TAB key to change the character size.
	Enter	This terminates the data entry and adds pseudocodes to your program which will generate the text you have typed on screen.
Adding a Softkey:	Cursor	Move the cursor below the last “PUT TEXT” Pseudocode in the list below the begin screen.
	F5	Select the Pseudocode list.
	S	Move to the “S” section.
	Cursor	Move the cursor to the Softkey.
	Enter Softkey number	Select a Softkey number between 1 and 10 (key 7 through 10 do not place text on screen).
	Enter Softkey text	Enter the 5 character label you wish to place above the softkey
Add 2 more softkeys in the same manner	Enter screen to go to	Enter the name of a screen that you want your program to jump to when the softkey is pressed.

Operation:	**Keystroke:**	**Description:**
Waiting for a softkey	Enter	Adds the Pseudocode to your program.
	F5	Select the Pseudocode list.
	S	Move to the "S" section
Building more screens:	Cursor	Move the cursor to the SOFTKEY WAIT Pseudocode.
	Enter	Adds the Pseudocode to your program
	Cursor	Move the cursor to one of the SOFTKEY (skey)...Pseudocodes.
	F2	Goto build a screen.
	Enter	Answer "YES" to add a new screen.
	Repeat steps above	Repeat the steps above (excluding the cursor positioning over the GOTO SCREEN BUILD), this will add the rest of the screens necessary to build a program. When done adding screens:

When done adding screens:

Operation:	**Keystroke:**	**Description:**
Save program:	ALT-F	File name
	Cursor	Select SAVE AS.
	Enter	You will be prompted for a "SAVE AS" file name.
	TAB	Enter the text entry area above the file list.
	Filename	Enter a valid DOS filename with a "BLD" extension (i.e. NEWFILE.BLD).
	Enter	Save the file.
Convert to BASIC:	F8	Build BASIC
	Enter	Answer "YES" to enter the file into the EDITOR.
	ALT-T	Transfer menu.
	Enter	Download.

Operation:	**Keystroke:**	**Description:**
Run program:	F1 (T-60)	Run the program on the Model T-60.

Screen Editor

This Chapter briefly describes the operation of the T-60's built-in screen editor. For a detailed description of the built-in screen editor, refer to the EMERSON EMC BASIC PROGRAMMING GUIDE. This editor allows the programmer to edit programs directly on the T-60's screen without a PC attached. This feature is most useful for program debug or for constructing and modifying short programs. We recommend that you utilize the ApplicationBuilder to generate more substantial programs. The ApplicationBuilder will shorten your development time, give you much more accurately generated BASIC code, and provide a means of storing, documenting and cataloging your programs.

Editor Capabilities

The built-in screen editor has a complete set of features which allow the programmer to develop programs directly on the T-60's screen. Operation of the editor requires the connection of an external PC-XT keyboard (see section "Getting Started"). Once the keyboard is connected, you can access every programmable function in the T-60. The editor capabilities center around the T-60's 8 line by 40 character display. It has a built-in 50 line scroll buffer which can be used to view sections of programs (or entire programs if they are short enough). The editor has the following feature set:

Line oriented editor - change a line by over-typing on it, press ENTER and it is modified in the BASIC program.

78 character line capability. The editor can edit lines longer than the screen width. If you keep typing after the first 40 characters, the editor will insert a "¦" character at the end of the current line and the beginning of the next line. You may continue typing on the next line. The "¦" characters do not consume space in your program memory.

Full cursor control: Up, Down, Left, Right, Page-Up, Page-Down, Home, End, Insert, Backspace, and Delete

Special functions: Control-Y stop program execution and return to the editor; Control-T Delete to end of line.

Active function keys:

F1	RUN	←	
F2	LIST		
F3	REMOT	←	(remote on/off *)
F4	CONT	←	
F5	AUTO		
F6	EDIT		

* Useful for uploading and downloading programs with programs other than the ApplicationBuilder. See the REMOTE Statement in the EMERSON EMC BASIC PROGRAMMING GUIDE.

NEW command erases the contents of the editor memory.

DELETE command erases sections of the program by line number.

RENUM allows the entire program or portions of the program to be renumbered.

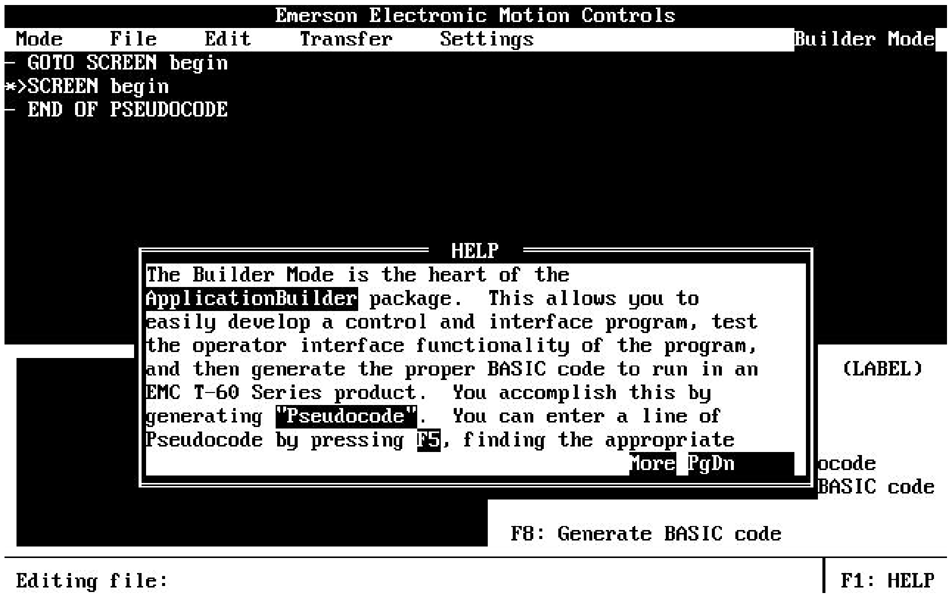
LIST lists all or sections of the program to the screen.

LLIST can list all or sections of the program to the printer.

Help Key

The T-60 is equipped with a powerful HELP feature. By pressing the HELP key on the T-60's front panel or F10 on the IBM keyboard while in the BASIC Interpreter Screen Editor (not running a program), the following screen appears:

Figure 12
Help Screen



```
                    Emerson Electronic Motion Controls
Mode    File    Edit    Transfer    Settings                Builder Mode
- GOTO SCREEN begin
*>SCREEN begin
- END OF PSEUDOCODE

                         HELP
           The Builder Mode is the heart of the
           ApplicationBuilder package.  This allows you to
           easily develop a control and interface program, test
           the operator interface functionality of the program,
           and then generate the proper BASIC code to run in an      (LABEL)
           EMC T-60 Series product.  You accomplish this by
           generating "Pseudocode".  You can enter a line of
           Pseudocode by pressing F5, finding the appropriate
                                                  More PgDn       ocode
                                                                  BASIC code

                                       F8: Generate BASIC code
 Editing file:                                                    F1: HELP
```

The HELP screen consists of three basic sections:

1. **TOP** of screen. Function keys F7, F8, and F9 scroll through the lists of available commands for the T-60. F7 generates a list of commands. By pressing the cursor keys, a command will be highlighted. Pressing Enter selects the command and displays the help syntax on the screen. F8 and F9 scroll through the list of commands in alphabetical order.

2. **MID** screen. 5 lines of text which describe the COMMAND that the cursor was placed on prior to pressing the HELP key, and an example of it's SYNTAX.

3. **FUNCTION** keys. Six function keys are active during the help system. These keys perform the following functions:

COM1	Displays the buffers and status of COM1.
COM2	Displays the buffers and status of COM2
SYS	Displays system related status.
I/O	Displays 8-bit I/O positions and allows the operator to change them.
LITE	Select LCD backlight auto shutdown, ON, or OFF.
ON	Leaves the backlight on at all times.
OFF	Leaves the backlight off at all times.

AUTO	Turns backlight off after 10 minutes if no activity occurs such as program execution or a key press. The backlight comes back on as soon as a key is pressed.
EXIT	Press this key to leave the HELP system.

Try the HELP system. Type GOTO and back the cursor up with the or - cursor keys until the cursor is underneath one of the characters of the GOTO statement. Press HELP. The help screen will display a brief description of GOTO. Press EXIT to return to the BASIC Interpreter.

The HELP key is treated differently when the BASIC Interpreter is running a program. When running a BASIC program, the HELP key becomes a function key, F10. For example, ON KEY (10) GOSUB T-60 will generate an interrupt subroutine call to location 1000 when the HELP key is pressed. Think of the HELP key as a “pre-labeled” function key, F10. By keeping track of the operational status of your program you can generate context sensitive HELP for your application. For example, by providing a variable named HELP which gets updated whenever the screen contents change, the subroutine which responds to the HELP key can examine this variable and print a specific message out to the user which instructs the user what to do at any given time.

Examine the demo programs supplied with the T-60. These programs make use of the function key interrupts, and in particular, the F10 or HELP interrupt.

Application Examples

This chapter demonstrates a few of the many applications possible with the T-60. Perhaps the best way to think of the T-60 is as an industrial controller with an integrated operator interface and very capable I/O. In many systems the T-60 can provide the entire system control. In others, intelligent controllers such as EMERSON EMC DX series positioning servo drives with integrated motor drivers perform the motion control while the T-60 performs the man-machine interface and overall orchestration. In all, the T-60 is as “smart” as you need to make it. The ApplicationBuilder and the EMERSON EMC BASIC can perform some powerful control functions.

Communicating with the Operator

Most applications require some sort of man-machine interface. During set-up it may be necessary to allow a significant number of parameters to be adjusted. During machine operation, it is often desirable to display status information and limit the operator’s ability to change crucial process parameters. The T-60 is extremely capable of providing this flexibility, and it is especially easy to program into the T-60.

Our first example program is NUMBER.BLD. Load the program into the ApplicationBuilder (if you are not familiar with the ApplicationBuilder, please refer to “ApplicationBuilder” section). NUMBER.BLD is REALLY simple... it asks for a numeric entry from the operator, and re-displays it on the screen in another location. Go ahead, try it.

Notice that the operator entry is performed in a calculator-like fashion. A default value is displayed on screen. The first key that you press clears the data entry area and allows you to edit data with the left and right arrow keys or the insert and delete keys. When you press enter, the data is stored. The PSEUDOCODE used to generate this function is the GET NUMBER pseudocode. The ApplicationBuilder utilizes the CALL NINPUT BASIC statement to construct this entry. The ApplicationBuilder simply calls this function to perform data entry.

The number you have entered is printed out on screen in three different character sizes. PUT NUMBER places regular size characters on screen. This function is performed by positioning the cursor with the POS command and using PRINT. PUT LARGE NUMBER and PUT HUGE NUMBER places large (20 characters by 4 lines) or huge (10 characters by 2 lines) on screen. These pseudocodes are implemented by the CALL BANNER BASIC function.

Performing data entry and data display with the ApplicationBuilder is about as easy as using a very basic word processor. Use GET NUMBER to get data and use PUT NUMBER or PUT TEXT to place information on the screen.

Serial Communications made Simple

The real power of the T-60 starts demonstrating itself when I/O is used. The ApplicationBuilder file SERIAL.BLD demonstrates simple serial communications between the T-60 and a serial device (in this case the ApplicationBuilder’s Terminal Emulator).

Load and convert SERIAL.BLD into a BASIC file (F8 function). Load it into the T-60. Now, enter the terminal emulator (ALT-M then type T). Press the RUN (F1) softkey on the T-60. You will be asked a question on your PC's screen. Answer it and press Enter. Watch what happens.

The ApplicationBuilder utilizes COM: PUT TEXT pseudocode to transmit serial data. Actually, COM: PUT TEXT is converted into the PRINT # BASIC statement to perform serial I/O. Data is gathered into a string variable and output directly to the port in a single instruction. The COM: INIT pseudocode generates a CONFIG # BASIC statement to initialize the I/O. The complexities of the CONFIG statement are hidden from the programmer with the simplicity of the pseudocode.

Parallel I/O Made Simple

If you thought Serial I/O was easy wait until you take a look at Parallel I/O. The T-60 has 8 parallel I/O lines built into it. The ApplicationBuilder has a host of I/O pseudocodes. To look at them, press F5 then press the I key. This will display a list of available I/O pseudocodes.

An example program which sequences I/O bits one at a time is on your ApplicationBuilder diskette: IO.BLD. Try loading it and running it. This program uses on-screen messages to tell you about itself.

EMERSON EMC DX Drive Demo Program

Please read the first part of the DX.-COM.BLD program. There are instructions on how the DX Amplifier must be setup to operate correctly.

This program (DX-COM.BLD) was written to demonstrate the capability of the T-60 to control a EMERSON EMC DX Drive. The T-60 starts the program at power on, and begins communicating with the DX Drive. It then allows the user to make simple moves, download sequences, and scale units. This program provides a good starting place for understanding how to communicate with the EMERSON EMC DX Drives.

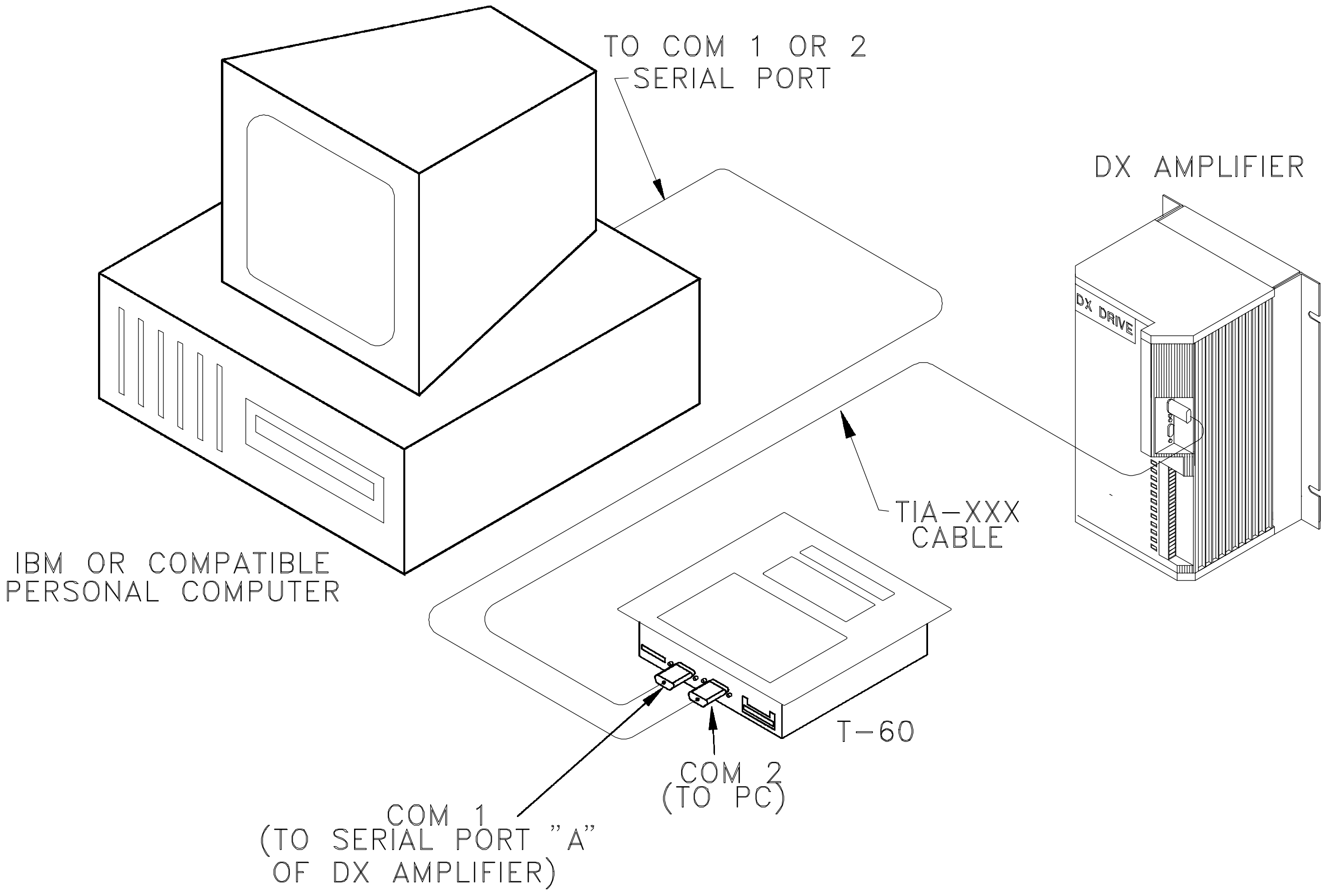
Installation

Connect the T-60 to the PC and to the DX Drive Amplifier as shown in Figure 17. T-60's COM 2 port connects to the PC's COM 1 or COM 2 port. This connection must be made with a NULL MODEM cable. The T-60's COM 1 port connects to the DX Drive Amplifier's serial "B" port.

You will need a serial communications cable (Model number TIA-XXX), to connect a DX Drive to the T-60. Such a cable can be obtained from EMERSON EMC in three standard lengths:

Part Number	**Length**
TIA-010	10 FT
TIA-025	25 FT
TIA-050	50 FT

Figure 13
Installation



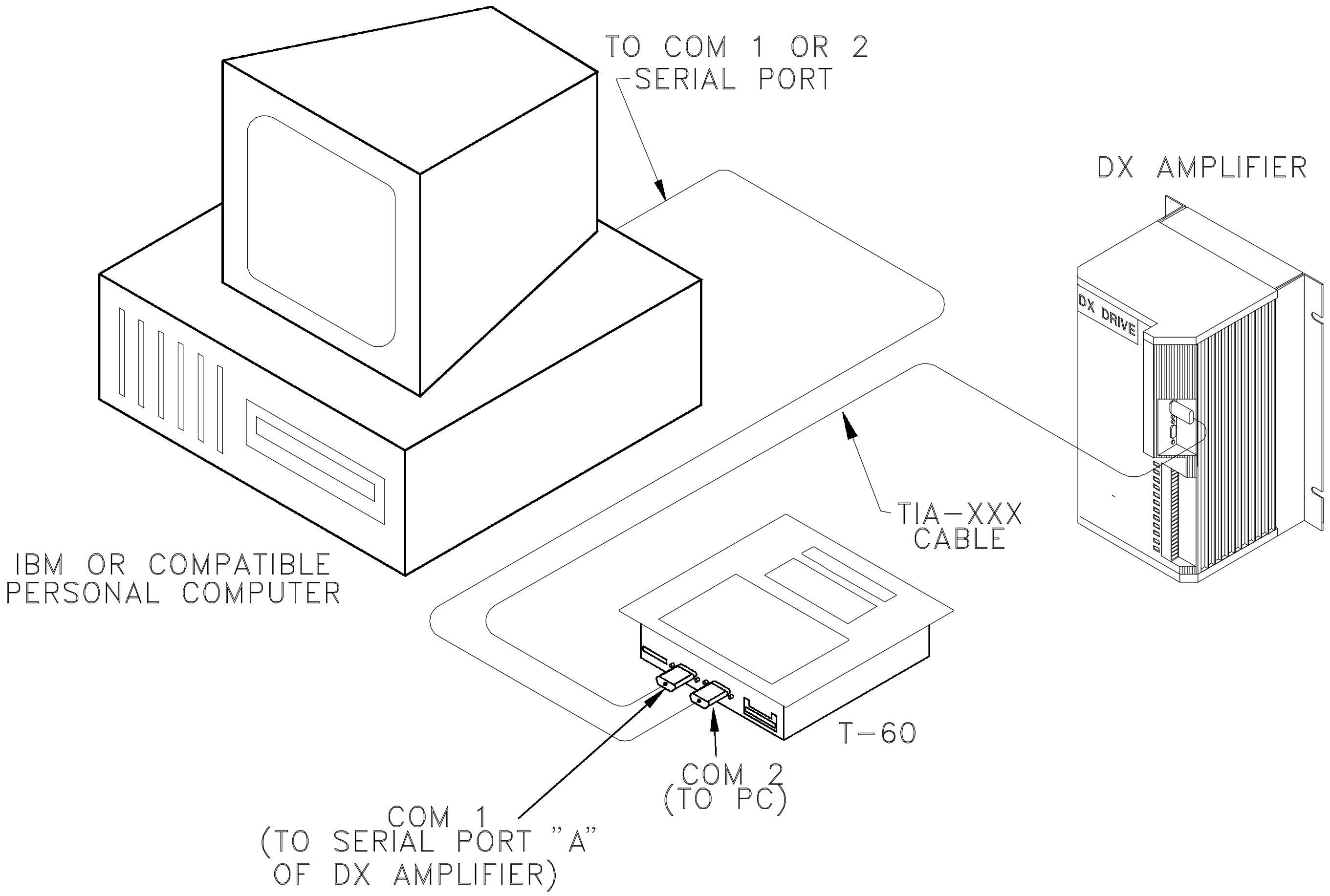
When power is turned on to the T-60 and the DX Drive, the ApplicationBuilder program will test for communication's integrity, and begin execution. If the T-60 is unable to establish communications, check the following items:

- Verify that the DX Drive Amplifier is on and in working order.
- A power-on sequence must be setup which automatically starts executing on power up.
- Verify the connections made between the drive and the T-60.

Using The T-60 With One or More DX Drives

Connecting the T-60 to the DX drive(s) is very simple when the proper cables are used. The Figure below illustrates the required cables and what they should be connected to. The PC's COM1 or COM2 is connected to COM2 on the T-60. Note that the cable used for this connection is a NULL MODEM cable, and can be obtained from EMERSON EMC. COM1 on the T-60 is connected to the DX drive via a TIA cable also available from EMC.

Figure 14
Installation - T-60 W/More Than One DX Drive



Baud rates and setup

The build software should be able to communicate with the T-60 without deviation from the default comm parameters. The DX drive serial parameters must be set up as desired along with the axis id. These parameters will be matched by the COM initialization pseudo-command provided by the builder software.

DX specific pseudo-commands

There are 2 pseudo-commands which are used only with the DX drives:

```
COM:   DX PUT COM port TEXT text STRING RESPONSE response
COM:   DX SELECT AXIS axis
```

These commands are fully described in the builder software and illustrated in the included example programs.

T-60 Basic Programming Language

This chapter gives an overview of the capabilities of the EMERSON EMC BASIC. The BASIC language integrated into the T-60 has been custom designed and optimized for speed of execution and ease of use with the specific hardware features of the T-60. A complete description of the syntax is available in the EMERSON EMC BASIC PROGRAMMING GUIDE. This guide is provided with the T-60 and is available from EMERSON EMC or your EMERSON EMC DISTRIBUTOR.

Variables, Constants, and Strings

The EMERSON EMC BASIC has a range of numeric and string variable and constant types. Variable names may be up to 8 characters long. The characters allowed in a variable name are letters, numbers, and the decimal point. The first character in the variable name must be a letter. Special type declaration characters are also allowed. The types and their dynamic ranges:

Type:	**Symbol:**	**Description:**
Float!	(or none)	Floating point numeric variables and constants. Positive or negative numbers represented in exponential form (similar to scientific notation). A floating-point constant consists of an optionally-signed integer or fixed-point number (the mantissa), followed by the letter E and an optionally signed integer (the exponent). The allowable range for floating-point constants is 0.8388607X10^{-19} to -0.8388607X10^{14}. For example: 235.988E-7 = .0000235988 2359E6 = 2359000000
Integer	%	Integer numeric variables and constants. Whole numbers between -32768 and +32767. They do not contain decimal points.
Double-Precision Integer	&	Double precision variables and constants. Whole numbers between -2147483648 and +2147483647. They do not contain decimal points.
String ‘	$	String variables and constants. A string can consist of up to 127 alphanumeric characters.The default type for a numeric variable name is single-precision.You should be very careful when making conversions between integer, single-precision, and double-precision integers variables, rounding errors may occur.

Double-precision integers are useful for moderately fast math functions. Many different types of machine controllers require double precision numeric ranges for their input. Double-precision integers were implemented primarily for this purpose.

Array Variables

An array is a group or table of values referenced by the same variable name. Each element in an array is referenced by an array variable that is a subscripted integer or an integer expression. The subscript is enclosed within parentheses. An array variable name has as many subscripts as there are dimensions in the array.

For example,

```
V(10)
```

references a value in a one-dimensional array, while

```
T(1,4)
```

references a value in a two-dimensional array.

The maximum number of dimensions for an array in EMERSON EMC BASIC is 16383. Arrays cannot have a size greater than 32767 bytes. i.e. A(8191), b%(16383), and s$(16383) are all valid. A(8192), b%(16384), and s$(16384) are all invalid sizes for arrays. See 6.2.4 for a description of memory space requirements.

Multi-dimensional arrays (more than one subscript separated by commas) are useful for storing tabular data. For example, an array dimensioned with DIM A(2,5) could be used to represent a two-row, five-column array such as the following:

Column	1	2	3	4	5
Row 1	10	20	30	40	50
Row 2	60	70	80	90	100

In this example, element A(2,3)= 80 and A(1,4)= 406.3

Arithmetic Operation

The following are the arithmetic operators recognized by EMERSON EMC BASIC. They appear in order of precedence.

Operator:	**Operation:**
^	Exponentiation
-	Negation
*	Multiplication
/	Floating-point Division
MOD	Modulus
+	Addition
-	Subtraction

Operations within parentheses are performed first. Inside the parentheses, the usual order of precedence is maintained. Two consecutive operators must be separated by parentheses.

Relational Operations

Relational operators let you compare two values. The result of the comparison is either true (-1) or false (0). This result can then be used to make a decision regarding program flow.

=	Equality	X = Y
<>	Inequality	X <> Y
<	Less than	X < Y
>	Greater than	X > Y
<=	Less than or equal to	X <= Y
>=	Greater than or equal to	X >= Y

The equal sign is also used to assign a value to a variable.

When arithmetic and relational operators are combined in one expression, the arithmetic is always performed first:

```
X+Y < (T-1)/Z
```

This expression is true if the value of X plus Y is less than the value of T-1 divided by Z.

Logical Operators

Logical operators perform tests on multiple relations, bit manipulation, or boolean operations. The logical operator returns a bit-wise result which is either true (not zero) or false (zero). In an expression, logical operations are performed after arithmetic and relational operations. The following table lists the availabe logical operators.

Operator:	**Operation:**
NOT	Logical negation
AND	Logical AND
OR	Logical OR
XOR	Logical exclusive OR

Just as the relational operators can be used to make decisions regarding program flow, logical operators can connect two or more relations and return a true or false value to be used in a decision. For example:

```
IF D<200 AND F<4 THEN 80
IF I>10 OR K<0 THEN 50
IF NOT P THEN 100
```

It is possible to use logical operators to test bytes for a particular bit pattern. For instance, the AND operator may be used to mask all but one of the bits of a status byte at a machine I/O port. The OR operator may be used to merge two bytes to create a particular binary value. The following examples demonstrate how the logical operators work:

Example	Explanation
63 AND 16=16	63=binary 111111 and 16=binary 10000, so 63 AND 16=16
10 OR 10=10	10=binary 1010, so 1010 OR 1010=1010(10)
NOT X = -(X+1)	The two's complement of any integer is the bit complement plus one.

Functional Operators

A function is used in an expression to call a predetermined operation that is to be performed on an operand. BASIC has intrinsic functions that reside in the system, such as SQR (square root) or SIN (sine).

BASIC also allows user-defined functions written by the programmer. See the DEF FN statement in the EMERSON EMC BASIC PROGRAMMING GUIDE.

The CALL instruction allows access to T-60 machine specific features such as special screen functions or option board functions. The CALL instruction may have optional parameters associated with it. Refer to the CALL instruction in the EMERSON EMC BASIC PROGRAMMING GUIDE.

String Operators

To compare strings, use the same relational operators used with numbers:

Operator	Meaning
=	Equal to
<>	Unequal
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

The BASIC Interpreter compares strings by taking one character at a time from each string and comparing their ASCII codes. If the ASCII codes in each string are the same, the strings are equal. If the ASCII codes differ, the lower code number will precede the higher code. If the interpreter reaches the end of one string during string comparison, the shorter string is said to be smaller, providing that both strings are the same up to that point. Leading and trailing blanks are significant.

For example:

“AA”	<	“AB”
“FILENAME”	=	“FILENAME”
“SMYTH”	<	“SMYTHE”
B$	<	“9/12/78” where B$ = “8/12/78”

String comparisons can also be used to test string values or to alphabetize strings. All string constants used in comparison expressions must be enclosed in quotation marks.

Strings can be concatenated by using the plus (+) sign. For example:

```
10 A$=”FILE”:B$=”NAME”
20 PRINT A$+B$
30PRINT “NEW” + A$+B$
RUN
FILENAME
NEW FILENAME
```

EMERSON Basic Statements, Commands, and Functions

The following is an alphabetized list of statements commands and functions available in the EMERSON EMC BASIC. The EMERSON EMC BASIC PROGRAMMING GUIDE contains a detailed description of the each of these commands. Eason Technology is constantly adding new capability to the T-60, please contact the factory or your Eason Technology Distributor for the latest copy of the EMERSON EMC BASIC PROGRAMMING GUIDE. Copies of the guide are also available on diskette. You can use the ApplicationBuilder Editor to load and search for the description of a command without having to search for a manual.

ABS	Absolute value	INKEY$	Read keyboard (returns string)
ASC	ASCII to number	INP	Input from 8-bit I/O
ATN	Arctangent	INPUT	Get user input
AUT	Auto line number	INPUT#	Input from comm port
BIN	Binary to decimal	INPUT$	String input
BIN	Decimal to binary	INSTR	Search for string
CALL	T-60 special function	INT	Truncate to whole number
CALL BANNER	Print large characters	IO24	Specify an I/O bit pattern to trap on
CALL BOX	Graphics box draw function	KEY	Function key statements
CALL GCLS	Clear graphics screen	KEY(n)	Function key interrupts
CALL HELP	Display text with help text	LABEL	Line label
CALL LINE	Draw graphics line	LEFT$	Left characters of a string
CALL NFORMAT	Formatted numeric string conversion	LEN	Length of string LINE
CALL NSETUP	Formatted numeric setup	INPUT	Input until Enter
CALL NKEY	Formatted numeric input	LINE INPUT	#Line input from comm port
CALL NINPUT	Formatted numeric entry	LISTDisplay	Lines on screen
CALL POINT	Return color of graphics point	LLIST	Print lines
CALL SCANKEY	Scan the keyboard return keycode	LOG	Natural log
CALL SET	Draw graphics pixel	LOCK	Secure program from edit,view
CHR$	Number to ASCII	LPRINT	Write to printer
CLEAR	Clear variables	LPRINT USING	Formatted LPRINT
CLS	Clear screen	MID$	Substring operations
COM(n)	Communications trapping	NEW	Clear program
CONFIG	Communications settings	OCT	Octal to decimal
CONT	Continue	OCT$	Decimal to octal
COS	Cosine	ON COM(n)	Interrupt on comm port
CSRCOL	Cursor row	ON IO24	Interrupt on I/O bit pattern
CSRROW	Cursor column	ON KEY(n)	Interrupt on function key
DATA	Data storage	ON TIMER(n)	Interrupt on timer
DEF FN	Define function	ON ERROR GOTO	Interrupt on error
DELAY	Millisecond delay	ON...GOSUB	GOSUB to list of line numbers
DELETE	Delete program lines	ON...GOTO	GOTO list of line numbers
DIM	Dimension array	OUT	Output to 8-bit I/O
EDIT	Edit program lines	OUT XOR	Logical exclusive OR to 8-bit I/O
END	End of program	OUT AND	Logical AND to 8-bit I/O
ERASE	Erase arrays	OUT MAP	Set/clear bits on 8-bit I/O
ERR	Error code	OUT OR	Logical OR to 8-bit I/O
ERL	Line number with an error	POS	Position Cursor
ERROR	Simulate error	POWER RESUME	Auto startup
EXP	Exponentiate	PRINT	Write to display
FIX	Truncate to whole number	PRINT USING	Formatted PRINT
FOR	FOR-NEXT loops	PRINT #	Write to comm port
NEXT	FOR-NEXT loops	PRINT # USING	Formatted PRINT #
FRE	Free Space	READ	Read data
GOSUB	Subroutine call	REM	Comment
GOTO	Jump to line number	REMOTE	Host computer control
HEX	Hexadecimal to decimal	RENUM	Update line numbers
HEX$	Decimal to hexidecimal	REPEAT	REPEAT-UNTIL loops
IF	Conditional Statement	UNTIL	REPEAT-UNTIL loops
INKEY	Read keyboard (returns number)	RESTORE	Re-read data statements

RESUME	Continue after error	STRING$	Multiple copies
RETURN	Exit GOSUB	TAB	Tab spaces
RIGHT$	Substring operations	TAN	Tangent
RND	Random number	TIME	Internal timer
RS422	RS422 port control	TIME$	Set/retrieve time
RUN	Start program	TIMER	Initialize timer interrupt
SGN	Get sign of number	TRACE	Trace ON/OFF
SIN	Sine function	VAL	String to number
SPACE$	Generate spaces	VER	Report software version
SQR	Square root	WHILE	WHILE-WEND loops
STOP	Halt program	WEND	WHILE-WEND loops
STR$	Convert to string		

64K Memory Option

The M02 option adds 32K bytes of storage to the T-60's battery-backed up memory to bring the total memory up to 64K bytes. Larger programs and more data can be stored in this additional memory. In addition, nonvolatile storage registers can also be used when this option is added. 480 numeric registers, floating point, fixed precision, and integers (in any combination), as well as 16 string registers (128 bytes each) are available for storing seldom-changed program constants and data. Power failure, program loading, variable clearing, or gosub stack clearing will not affect the data stored in these registers.

You can tell if the M02 option has been installed in your unit by pressing the HELP key when the T-60 is not running a program (in command mode). By pressing the SYS key, the amount of memory installed will be indicated on the screen. It should read 64K. The amount of free memory will vary depending upon data and programs loaded. With no data or no programs loaded, the free memory should be 52735 bytes.

The M02 option adds access to a pre-dimensioned array called NVOL. NVOL can contain up to 512 floating point, short integer or long integer variables. In addition, the M02 option adds access to a pre-dimensioned string array called NVOL$. NVOL$ allows access of up to sixteen 128 byte string array variables. The M02 option is field installable.

Adding Memory Options to the T-60

Follow these steps to upgrade the T-60 to 64K of RAM:

1. Save the contents of the program memory. Use a PC and the ApplicationBuilder program to do this. Changing the internal memory will require resetting the system, thereby erasing the contents of all memory.

2. Remove power connections to the unit.

WARNING

Disconnect power from the T-60 before attempting to install ram. You may cause damage to the unit if the power is not disconnected.

3. Carefully unscrew the top two screws and each of the screws on the side of the unit.

4. Lay the unit on its back on a soft, clean surface.

Check the serial number of your T-60. If it begins with “ETI”, please go to section B.Otherwise, proceed here with section A.

Section A

1. Remove the front panel by carefully lifting it off the unit and disconnecting the two pin connector for the backlight cable (J6, see diagram).

2. Position the front panel to the side of the unit, being careful not to stress the internal cables.

3. To install an MO2 option, insert a 256Kbit RAM (120 nsec, JDEC 32 pin) in location U9. Be sure that PIN 1 (location on RAM with notch or dot) is facing down (towards the center of the unit). Note carefully the position of U9 and the orientation of the RAM in the diagram. Check carefully for bent or damaged pins.

WARNING

Improper orientation of U9 will destroy the RAM, and possibly cause damage to the unit.

4. Re-connect the backlight cable connector to J6 (see diagram) and replace the front panel and screws.

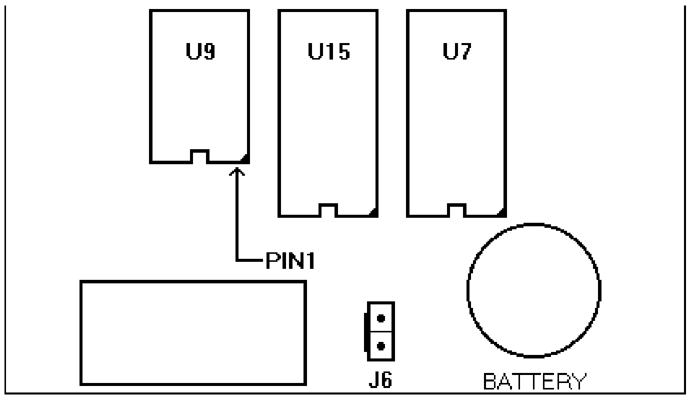
5. Connect an IBM KEYBOARD to the KEYBOARD connector. Make sure that the keyboard is a type which is compatible with an IBM PC XT (not AT).

6. Re-connect power to the unit. Press the CTRL, ALT and DEL keys on the IBM KEYBOARD simultaneously. This will reset the unit and normal operation will begin.

7. Check the installation by pressing HELP on the T-60 followed by F4 (SYS). The memory option should show 64K.

8. If the memory option only shows 32K, or the display does not come up displaying the sign-on message, REMOVE POWER IMMEDIATELY. Remove the cover and recheck the installation of U9.

Figure 15
Top of Board Inside T-60



Section B

1. Remove the front panel by carefully lifting it off the unit and disconnecting the two pin connector for the backlight cable (J10, see diagram).

2. Position the front panel to the side of the unit, being careful not to stress the internal cables.

3. To install an MO2 option, insert a 256Kbit RAM (120 nsec, JDEC 32 pin) in location U9. Be sure that PIN 1 (location on RAM with notch or dot) is facing down (towards the center of the unit). Note carefully the position of U9 and the orientation of the RAM in the diagram. Check carefully for bent or damaged pins. If you are installing sn MO3 option, repeat the procedure in locations U45 and U46.

WARNING

Improper orientation of U9, U45, or U46 will destroy the RAM and possibly cause damage to the unit.

4. Re-connect the backlight cable connector to J10 (see diagram) and replace the front panel and screws.

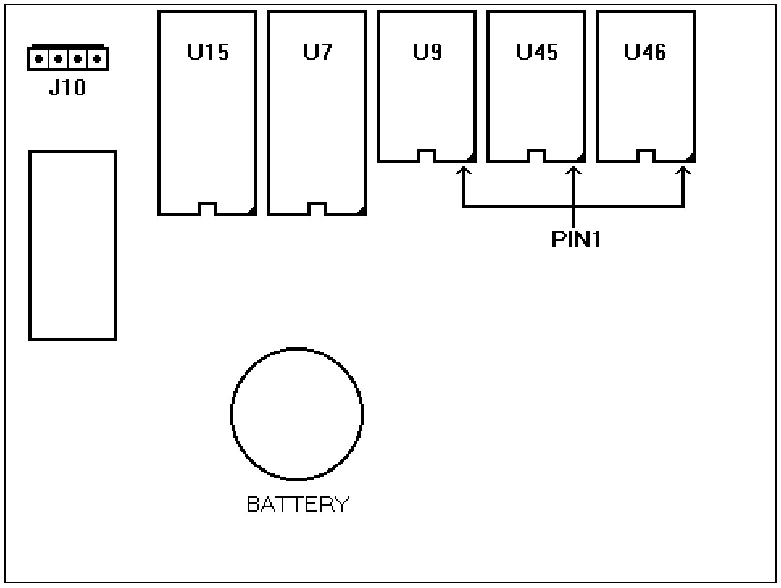
5. Connect an IBM KEYBOARD to the KEYBOARD connector. Make sure that the keyboard is a type which is compatible with an IBM PC XT (not AT).

6. Re-connect power to the unit. Press the CTRL, ALT and DEL keys on the IBM KEYBOARD simultaneously. This will reset the unit and normal operation will begin.

7. Check the installation by pressing HELP on the T-60 followed by F4 (SYS). The memory option should show 64K if you installed an MO2; 128K if you installed an MO3.

8. If the memory option only shows 32K, or the display does not come up displaying the sign-on message, REMOVE POWER IMMEDIATELY. Remove the cover and recheck the installation of U9, U45 and U46.

Figure 16
Top of Board Inside the T-60



Changing ROMS in the T-60

Follow these steps to change a program ROM in the T-60:

1. Save the contents of the program memory. Use a PC and the ApplicationBuilder program to do this.

2. Remove power connections to the unit.

3. Carefully unscrew the top two screws and each of the screws on the side of the unit.

4. Lay the unit face down on a soft, clean surface.

5. Remove the rear cover by carefully lifting it off the unit.

6. Position the cover to the side of the unit, being careful not to stress the internal cables.

7. Remove the ROM located at the top of the main board (top being closest to the display). Note the orientation of PIN 1 on the device (see diagram).

8. Insert the new ROM in location U15. Be sure that PIN 1 (location on ROM with notch or dot) is facing down (towards the center of the unit). Check carefully for bent or damaged pins.

WARNING

Improper orientation of U15 will destroy the ROM and possibly cause damage to the unit.

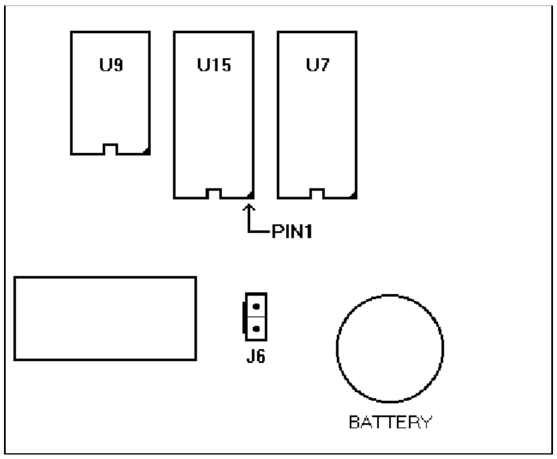
9. Re-attach the backlight cable (2 pin cable) to J6 (near U15, see diagram).

10. Replace the cover and screws.

11. Connect an IBM KEYBOARD to the KEYBOARD connector. Make sure that the keyboard is a type which is compatible with an IBM PC (not AT).

12. Re-connect power to the unit. Press the CTRL, ALT, and DEL keys on the IBM KEYBOARD simultaneously. This will reset the unit, and normal operation will begin.

Figure 17
Top of Board Inside the T-60



Changing the Battery in a T-60

Follow these steps to change the battery for the non-volatile RAM in a T-60:

1. **Save the contents of the program memory.** Use a PC and the ApplicationBuilder program to do this. Changing the battery may reset the system if you wait too long to change it or don't change it quickly, thereby erasing the contents of all memory.

2. Remove power connections to the unit.

WARNING

Disconnect power from the T-60 before attempting to change the battery. Hazardous voltages exist inside the unit with power connected. Personal injury or death may result if the power is not disconnected.

3. Carefully unscrew the top two screws and each of the screws on the side of the unit.

4. Lay the unit face down on a soft, clean surface.

5. Remove the rear cover by carefully lifting it off the unit.

6. Position the cover to the side of the unit, being careful not to stress the internal cables.

7. To remove the old battery, insert a small screwdriver between the edge of the battery and the battery holder near the battery clip (see diagram). Push gently and the battery should slide out.

Figure 18
Battery Replacement

8. Before installing the new battery, press down on the battery clip to make sure that there will be good retention force when the battery is in place. Also make sure that all contact surfaces are clean. As noted on the battery clip, make sure that the positive (+) side of the battery is up. Slide the battery under the clip and into the battery holder. Make sure that the battery is held firmly in place.

WARNING

Improper orientation of the battery will not allow the battery backed RAM to function and your memory will cease to be non-volatile. However, this will not damage the battery or the controller; just turn the battery over and continue with the installation procedure.

9. Replace cover and cover screws.

10. Reconnect power to the unit.

11. If you notice any problems that can't be explained by the battery being installed upside down (no non-volatile memory), disconnect power and connect an IBM KEYBOARD to the KEYBOARD connector. Make sure that the keyboard is a type which is compatible with an IBM PC XT (not AT).

12. Re-connect power to the unit. Press the CTRL, ALT, and DEL keys on the IBM KEYBOARD simultaneously. This will reset the unit, and normal operation will begin. Call Emerson EMC if any problems persist.

Real Time Clock

The CLK option adds a battery backed real time clock to the T-60. Once installed you can set the date, day of the week and time with a single statement: CALL WRCLOCK. You can read the day, day of the week, and the time with another single statement: CALL RDCLOCK. Refer to the EMERSON EMC BASIC USERS GUIDE for the programming syntax for these new commands. The CLK option's battery is integrated into a removable module which has a sealed internal lithium battery which is designed to last 10 years. The CLK option is field installable.

Adding the Real -Time Clock to the T-60

Follow these steps to add a real-time clock (CLK option) to the T-60:

1. Save the contents of the program memory. Use a PC and the ApplicationBuilder program to do this.

2. Remove power connections to the unit.

WARNING

Disconnect power from the T-60 before attempting to install a CLK Option.

3. Carefully unscrew the top screws and each of the screws on the side of the unit.

4. Lay the unit on its back on a soft, clean surface.

5. Remove the front panel by carefully lifting it off the unit and disconnecting the 2 pin connector for the backlight cable (J6, see diagram).

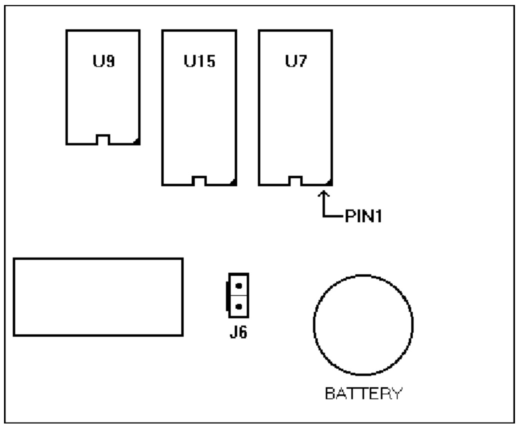
6. Position the cover to the side of the unit, being careful not to stress the internal cables.

7. Insert the CLK option in socket U7. Be sure that PIN 1 (location on the CLK with notch or dot) is facing down (towards the center of the unit, see diagram). Check carefully for bent or damaged pins.

WARNING

Improper orientation of the CLK will destroy it, and possibly cause damage to the unit.

Figure 19
Top of Board Inside the T-60



8. Reconnect the backlight cable connector and replace the front panel and screws.

9. It may be necessary to set the real-time clock before it can be used. Refer to the BASIC User’s Manual for instructions on how to set the clock.

Event Driven Software

The EMERSON EMC T-60 Event Driven Software allows the T-60 to react quickly to external events, without taxing the overhead of the imbedded BASIC programming language. The Event Software allows the T-60 to scan an input, react to it, and immediately output data to the A/D's or the 8 bit I/O port.

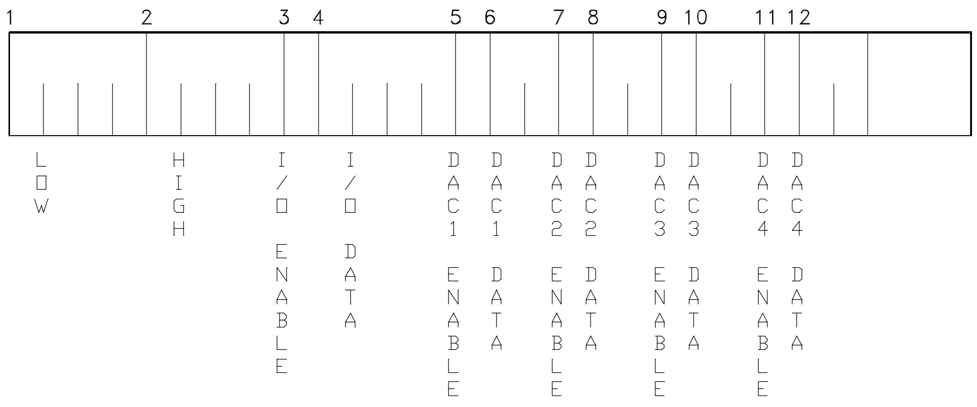
The input or source can be derived from bits 0 through 10 of the 8 I/O, an analog input voltage from an option module (optional), or from a counter on the counter option module (also optional). The Event Software is table driven. The program enters data into multiple rows in this table (see Figure). Each of these rows is called a “Schedule”. When the Event Manager software is activated, it reads data from the source and compares it to low and high compare values that were programmed into each schedule. When the data read from the source is within the range specified for a given schedule, the Event Manager automatically outputs I/O values, optional A/D values, and can GOSUB to a BASIC subroutine. The BASIC subroutine can determine which event has occurred by reading Event Manager data. Note that the I/O operations are optional. For example, an individual schedule can be programmed to only output one analog value, leaving the I/O and the other analog values unchanged.

If the source is from I/O or an optional analog input, the comparisons are performed with 16 bit integer mathematics, giving the greatest degree of flexibility possible (overlapping events and 32 possible schedules). If the source is from the optional counter, the comparisons are performed with 32 bit integer mathematics, limiting the events to non-overlapping regions and 16 possible schedules. The M02 option is required for the operation of the EVENT DRIVEN SOFTWARE.

Each schedule is comprised of a 32 byte record. There are up to 32 schedules available (16 for the optional counter). Individual elements within the table represent a comparison values which are specified as “LOW” and “HIGH”. If the event data falls within the range specified by LOW and HIGH, the previous event data is compared to see if the data was previously within the same region. If they are different (meaning a change has been encountered), a change of events has occurred. If the enable bytes for I/O and DAC's are true (greater than 0), the corresponding I/O or DAC is updated. If the BASIC interrupt system has been enabled to look for changes with the ON EVENT and EVENT ON statements, a BASIC interrupt can occur once every event change.Schedule 0 represents the left-over or unclaimed regions. There may be many unclaimed regions, or none. By programming schedule 0, you may set up data default output data which will always be refreshed when the event system leaves a specific event, and has no other specified event to go to.

Event Data Elements:

Figure 20
Event System - One Schedule Element



The EVENT SYSTEM is not enabled until the EVENT ON statement is issued. EVENT ON may be used without a corresponding ON EVENT statement, thereby avoiding the use of the T-60's interrupt system. See the example in the end of this section for the proper use of the EVENT ON syntax.

Hardware Reference

Electrical /Mechanical Specifications

Memory:	32K bytes battery backed RAM standard, 64K bytes optional
Communications:	Two RS232C ports, XON / XOFF handshaking capability COM 1 is configurable as RS232, RS422, or RS485. Optically isolated.
I/O:	8 bit parallel I/O; optically isolated -0.5 to +30V input, pulled up to +5V by 22K resistor Open collector output, sink 30 mA Industry standard PB-8 compatible.
Keyboard Connector:	IBM-PC XT compatible
Display:	High contrast Super-Twist LCD (liquid crystal display)
Size:	8 lines by 40 characters
Characters:	5 x 8 dot matrix; 0.095 x .15 (2.40 mm x 3.84 mm)
Dot size:	0.019" x 0.019" (0.48 mm x 0.48 mm)
Backlight:	Electro-luminescent panel with “auto-shutdown” feature Graphics capability.
Keyboard:	30 large, full-travel keys, rubber elastomer type
Help function:	Context sensitive HELP key available
Functions:	9 programmable function keys Cursor, insert, delete, alpha, and print keys
Power:	9V to 30V DC 6 Watts maximum. 1 Amp power-on inrush current.
Weight:	4.2 lbs (1.9 Kg)
Environment:	+32 to +122° F (0 to +50° C) 5% to 95% relative humidity (non-condensing) Designed to NEMA 4 (water-tight) and NEMA 12 (dust-tight) specifications.
Mounting:	Panel mount or flat surface mount (hardware included) Rugged, cast front housing
Physical size:	8.5"H X 8.5"W X 2.44"D

Connector Pin-Out Specifications

Power / RS422 Connector:

Pin	Function:
8	SHIELD
7	TX A
6	TX B
5	RX A
4	RX B
3	LOGIC GROUND
2	DC IN +
1	DC IN -

I/O Connector:

Pin:	Function:
19	+5V *
17	I/O 0
15	I/O 1
13	I/O 2
11	I/O 3
9	I/O 4
7	I/O 5
5	I/O 6
3	I/O 7
1	UNUSED

* (100 MA Load Maximum) All even numbered pins are connected to logic ground.

Keyboard Connector:

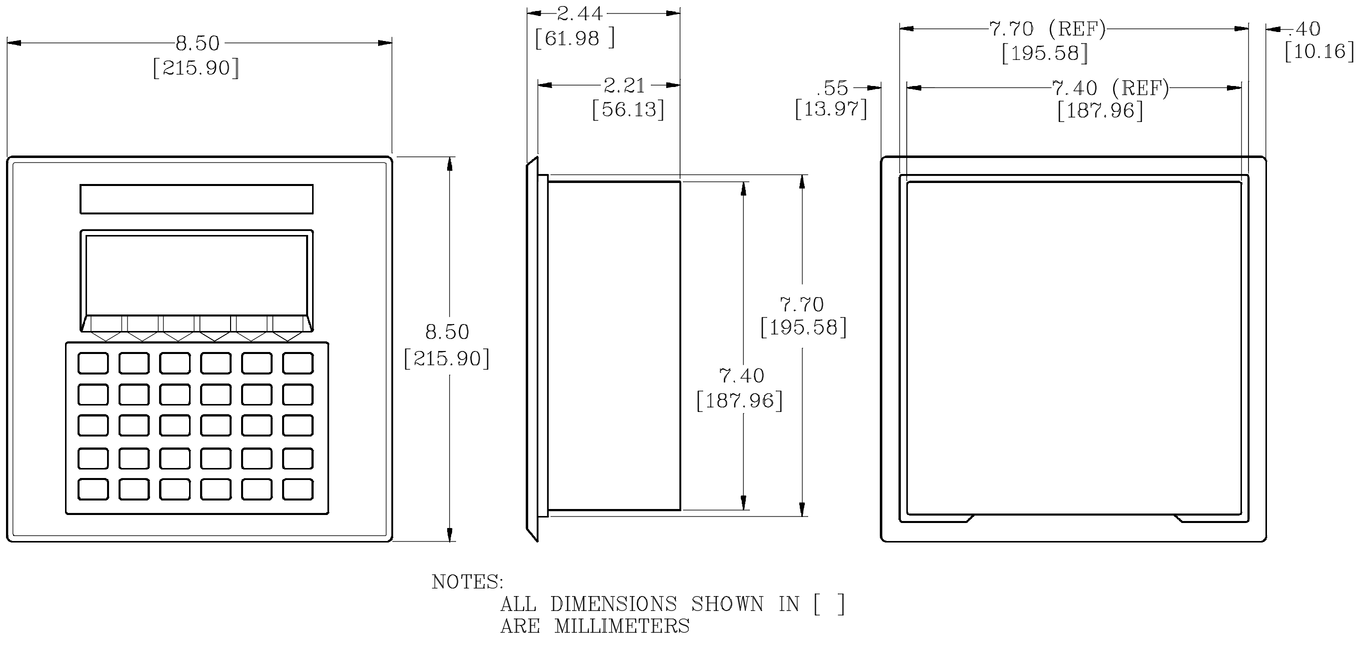
Pin:	Function:
1	KEYBOARD CLOCK
2	KEYBOARD DATA
3	KEYBOARD RESET
4	+5V
5	GROUND

COM1 and COM2 Pin-Outs:

Pin:	Connection:
HOUSING	SHIELD
2	RS232 RECEIVE (INPUT)
3	RS232 TRANSMIT (OUTPUT)
5	SIGNAL GROUND

All other pins not connected.

Figure 21
Physical Dimensions



PLC Interface Commands

Introduction

The EMERSON EMC T-60 Series PLC Interfaces are designed to make many of the individual intricacies of the various PLC Interfaces transparent to the user. That is to say that reading a register in a Modicon PLC uses the same procedure as reading a register in an Omron PLC. This allows a given program to be written for one PLC and then used with another PLC just by changing the PLC Interface option in your T-60 Series unit. This assumes, of course, that both PLC's have equivalent functionality.

For example, Modicon and TI treat analog I/O very differently (Modicon treats analog I/O as simply another register, whereas TI has separate memory locations for registers and analog I/O) so they are accessed differently via the T-60 Series.

Three "calls" are the heart of the PLC communications. They are CALL PLCINIT, CALL PLCREAD, and CALL PLCWRITE. These calls can be used in any reasonable location within a basic program running in the T-60 or T-61 with a PLC Interface option.

This allows the use of the data in a register, bit status, analog I/O value, or any other applicable data available from a PLC in a process or interface program in the T-60 Series device.

For most of the PLC Interface Options, the PLC Interface is on COM1 of the T-60 Series device. COM2 retains its default status. The -MOD, -PL5, and -SL5 Options have the ability to communicate to the PLC from either COM port. In effect, this allows you to hook two PLC's to each T-60 or T-61. In a T-60 , COM2 is still standard RS232C. In a T-61, COM2 can still be configured to be RS232C, RS422, or RS485.

This allows the integration of a PLC and virtually any other serial communications device. For example, you can connect a PLC to COM1 and another device that uses serial communication (e.g., motion controllers, PC's, serial displays, temperature controllers, etc.) to COM2 and communicate to all devices via the T-60 Series interface. In effect, the T-60 Series acts as an Operator Interface ***and*** an ASCII/BASIC module in one.

The tabel below shows the current PLC interface options.

Table 3
Current PLC Interface Options

Interface Option	PLC Protocols Supported	Software Version
-MOD	Modbus (Modicon)	20-0000X-02-X.XX
-GE9	GE Series 90 (GE Fanuc)	20-0000X-03-X.XX
-TI3	TI305 (GE Series 1) & TI405	20-0000X-04-X.XX
-TI5	TI505 (Siemens/TI)	20-0000X-05-X.XX
-PL5	Allen Bradley PLC-5	20-0000X-06-X.XX
-OM1	Omron Hostlink	20-0000X-07-X.XX
-PL2	Allen Bradley PLC-2	20-0000X-08-X.XX
-SL5	Allen Bradley SLC 500	20-0000X-09-X.XX
-ID1	IDEC FA-1J\FA2-J	20-0000X-10-X.XX
-MFX	Mitsubishi FX	20-0000X-11-X.XX
-SQD	Square D SY/MAX	20-0000X-13-X.XX+

As mentioned above, the PLC Interfaces all work in a similar, if not identical, fashion (the -PL5, -SL5, -ID1, -MFX, and -SQD interfaces are more unique than any of the others - refer to the PLC SPECIFIC sections for detailed information on how to use these interfaces).

Initialization of the PLC Interface (setting the proper communication parameters and verifying the establishment of the link between the PLC and the T-60 or T-61) **always** occurs with the CALL PLCINIT command. This command need only be issued once, usually during the initialization portion of your program. Any data that you wish to get from the PLC is retrieved by the CALL PLCREAD command. Data that you wish to write in a register, memory locations that you wish to define, or I/O bits you wish to set are all effected with the CALL PLCWRITE statement.

The following section describes the CALL PLCINIT, CALL PLCREAD, and CALL PLCWRITE commands in general. Most of the PLC Interfaces developed so far use the same arguments for similar functionality, no matter which Interface is installed. Exceptions are noted in the tables. More specific information follows in sections particular to each Interface Option.

Refer to the section for your PLC to see if the “normal” implementation for a given function is different for you PLC. See the PLC and SLC specific sections on the -PL5 and -SL5 Options for the CALL PLCREAD and CALL PLCWRITE commands for the Allen Bradley PLC-5 and SLC 500 PLC’s as they are significantly different from the rest of the interface options.

CALL PLCINIT Statement

Purpose:

This command is used to initialize communication with a given PLC of a given type. CALL PLCINIT must be issued prior to any other communication (reading or writing) to the specified PLC. CALL PLCINIT does auto-baud rate detect for some PLC's and sets up specific communications parameters for others. See the section specific to your Interface Option for information regarding the initialization parameters for that particular system. If the CALL PLCINIT command fails, the T-60 Series unit will break your program and generate the error message "PLC link not established." If this occurs, refer to the individual section for your option to make sure that the connections and communications configurations are correct. If they are OK, check to make sure that you do not have a password in your PLC that is not allowing access to the programming port.

Syntax:

CALL PLCINIT(*id*,*cmd*)

Comments:

id specifies the address of the PLC that you wish to initialize. Some PLC's allow addressing of multiple PLC's, others do not. Refer to your PLC Operator's manuals for information regarding capability and implementation.

Table 4
PLC Protocol Interface Options

Interface Option	Cmd	PLC Protocol
-MOD	T-60 Series COM port: 1 or 2	Modbus COM1
-GE9	1	GE Series 90
-TI3	Specific PLC Model: 315, 325, 330, 425, 435	TI305/405
-TI5	1	TI505
-PL5	T-60 Series COM port: 1 or 2	Data Highway Plus TM (DF1)
-SL5	T-60 Series COM port: 1 or 2	DH-485 TM (DF1)
-OM	T-60 Series COM port: 1 or 2	Host Link
-PL2	T-60 Series COM port: 1 or 2	PLC-2 Programming Port
-ID1	See IDEC Section	See IDEC Section
-MFX	See Mitsubishi Section	See Mitsubishi Section
-SQD	See Square D Section	See Square D Section

cmd specifies the type of PLC protocol that you are initiating. This can vary depending upon PLC manufacturer, and possibly model, according to the following table:

Examples:

10 CALL PLCINIT(1,1)
Establishes communication with a Modbus, GE Series 90, PLC-5, SLC-500, Omron Hostlink, or TI505 PLC (depending on the installed interface option) with an ID (address) of 1.

10 CALL PLCINIT(1,435)
Establishes communication with a TI435 PLC with an ID (address) of 1 (when the -TI3 option is installed).

10 CALL PLCINIT(1,2)
Establishes communication with a Modicon, AB PLC-5, or SLC-500 with an ID (address) of 1 on COM 2.

CALL PLCREAD Statement

Purpose:

This command is used to read the value(s) in a PLC's registers, the status of bits, or any other accessible memory location within the PLC. **Refer to the -PL5 and -SL5 for information on how to use this command with the Allen-Bradley PLC-5 and SLC-500 processors.**

Syntax:

CALL PLCREAD*(id,cmd,start address,# ofregisters/bits,variable/array)*

Comments:

id specifies the address of the PLC from which you wish to read data. This number is usually 1 when interfacing to one PLC.

cmd specifies the read operation you wish to perform. See the table below for the read operation possibilities.

start address is the starting address of the bit(s) or register(s) you are interested in reading. In the case of the PLC interfaces implemented so far, this does not include a data type specifier (%, $, !, etc.)

Table 5
Read Operation Options

MOD	GE9	TI3	TI5	OM1	PL2	Cmd	Function	Typical Data/Response*
•	•	•	•	•		1	Read PLC CPU Status	2 words; *ID, run status*
•	•	•	•	•	•	2	Read Discrete Input Status	1 word per 16 bits
•	•	•	•	•	•	3	Read Discrete Output Status	1 word per 16 bits
•	•	•	•	•	•	4	Read Register	1 word per register
•						5	Read Input Register	1 word per register
•						6	Quick Status	1 word with 8 bits
	•		•			7	Read Analog Inputs	1 word per register
	•		•			8	Read Analog Outputs	1 word per register
				•	•	9	Read Discrete Internals/Coils/Relays	1 word per 16 bits
				•		10	Read Holding Relays	1 word per 16 bits
				•		11	Read Auxilary Relays	1 word per 16 bits

*The Data/Response format described here is typical. Refer to the section regarding your specific interface for

# of registers/bits is the number of consecutive registers, memory locations, or bits that you wish to read. Most PLC interfaces only allow you to read one at a time, and for many applications that is all you will need to read. If this is the case, *# of registers/bits* will be 1. If you wish to read more than one register or memory location at a time, this number will be the number of consecutive registers or locations you wish to read. If you are reading the status of more than one I/O bit, this number will be the number of consecutive bits you wish to read.

variable/array is the variable name or array name where you wish to store the data you are reading. This variable **MUST** be a short integer (% variable). If the value of *# of registers/bits* (see above) is 1, this will be a variable expression. If you are reading registers or memory locations that are stored as words (*cmd* = 1, 4, 5, 6, 7, 8) and the *# of registers/bits* is greater than one, this will be a short integer array (make sure you properly dimension the array prior to using it). If you are reading the status of more than one, but less than 16 I/O bits, you will be reading one word, and therefore will need to use a single short integer. If you are reading more than 16 bits, you will need to use a short integer array. The dimension of the array variable will be the next integer greater than the desired number of bits divided by 16. For example, if you wish to read the status of bits 1-24 you will need to dimension your array to at least two since 24/16 = 1.5 and two is the next greater integer. Remember, any time you are reading more than one word of data (more than one register or more than 16 bits) you must use a **dimensioned short integer array.**

Examples:

```
10 DIM stat%(2)
20 CALL PLCREAD(1,1,0,2,stat%(1))
```

This command returns the ID of the PLC as stat%(1) and the current run status as stat%(2) from the Modbus or GE Series 90 PLC with an id (address) = 1.

```
10 DIM regdat%(10)
20 CALL PLCREAD(1,4,16,5,regdat%(4))
```

This command returns the contents of register 16 in regdat%(4), register 17 in regdat%(5), register 18 in regdat%(6), register 19 in regdat%(7), register 20 in regdat%(8), from any of the PLC’s currently implemented.

```
10 CALL PLCREAD(1,5,47,1,regdat1%)
```

This command reads the data in input register number 47 in the Modbus PLC with id (address) = 1 and stores it in the variable regdat1%

```
10 CALL PLCREAD(1,7,3,1,analog1%)
```

This command reads the value of analog input number 3 in a GE Series 90 or TI 505 PLC with address = 1 and stores it in the variable analog1%

CALL PLCWRITE Statement

Purpose:

This command is used to write value(s) to a PLC's register(s), memory location(s), or to force one or more output bits in a PLC. **Refer to the -PL5 and -SL5 for information on how to use this command with the Allen-Bradley PLC-5 and SLC-500 processors.**

Syntax:

CALL PLCWRITE(*id,cmd,start address,# of registers/bits,expression/variable/array* **)**

Comments:

id specifies the address of the PLC to which you wish to write. This number is usually 1 when interfacing to one PLC.

cmd specifies the write operation you wish to perform. See the table below for the write operation possibilities:

start address is the starting address of the bit(s) or register(s) you are interested in writing. In the case of the Modbus, this does not include a data type specifier (%, $, !, etc.)

Table 6
Write Operation Options

MOD	GE9	TI3	TI5	OM1	PL2	Cmd	Function	Typical Data*
		•				1	Clear Status	Any 5 words
•	•	•			•	2	Write Input Bit(s)	16 bit words
•	•	•	•	•	•	3	Write Output Bit(s)	16 bit words
•	•	•	•	•	•	4	Write to a Register	1 word per register
						5	N/A	N/A
						6	N/A	N/A
						7	N/A	N/A
	•		•		•	8	Write to Analog Output(s)	1 word per output
	•	•	•	•	•	9	Write to Discrete Internals/Coils/Relays	16 bit words
				•		10	Write to Holding Relays (bits)	16 bit words
				•		11	Write to Auxilary Relays (bits)	16 bit words

# of registers/bits is the number of consecutive registers, memory locations, or bits that you wish to set. Many PLC Interfaces only allow you to set one at a time, and for many applications that is all you will need to read. If this is the case, *# of registers/bits* will be 1.If you wish to set more than one register or memory location at a time, this number will be the number of consecutive registers or locations you wish to set. If you are writing the status of more than one I/O bit, this number will be the number of consecutive bits you wish to set.

expression/variable/array is the expression, variable or array data you wish to write to the PLC's I/O, registers, or other memory locations. If the value of *# of registers/bits* (see above) is 1, this will be an expression or a variable. If the number of words is greater than one, this must be an array (make sure you properly dimension the array prior to using it).If you wish to write the status of up to 16 I/O bits, you will be writing one word, and therefore will need to use an expression or discrete variable name. If you wish to set more than 16 consecutive bits, you will need to use an array. The dimension of the array variable will be the next integer greater than the desired number of bits divided by 16. For example, if you wish to set the status of bits 1-24 you will need to dimension your array to at least two since 24/16 = 1.5 and two is the next greater integer. Remember, any time you are writing more than one word of data (more than one register or more than 16 bits) you must use a **dimensioned array variable.**

Examples:

```
10 DIM stat(10)
20 CALL PLCWRITE(1,1,1,5,stat(1))
```

This command clears the status in a TI305/405 PLC

```
10 CALL PLCWRITE(1,3,12,5,13)
```

This command writes a 1 to outputs 12, 14, and 15, and a 0 to outputs 13 and 16 in a PLC with an id (address) = 1. Note that 13 represents the binary "01101" which is the bit pattern desired.

```
10 DIM newdat%(10)
20 CALL PLCWRITE(1,4,5,2,newdat%(4))
```

This command writes the value of newdat%(4) to register 5 and newdat%(5) to register 6 in a PLC with an id (address) = 1.

```
10 DIM anout%(10)
20 CALL PLCWRITE(1,8,1,2,anout%(1))
```

This command writes the value of analog1% to analog output 1 and to analog output 2 in a GE Series 90 or TI505 PLC with address = 1

PLC Specific Information

In general, PLC's from various manufacturers behave similarly. There are, however, some "quirks" that need to be mentioned. Also, various PLC's deal with Operator Interfaces in various ways. Some allow access to the registers, memory locations, and discrete I/O through the CPU's programming port, while others require some sort of communication module to be added to allow the CPU and Operator Interface to interact properly. This section deals with the criteria specific to each of the T-60 Series PLC Interface Options — command variances, communication hardware, cabling, etc.

Table 7

Option	PLC's Supported	Protocol	Communicate via	Com Parameters
-MOD	Any Modicon PLC Supporting Modbus: Micro 84, 484, 584, 184/384, 884, 984/381, etc.	Modbus	Programming Port	Auto-Detect
-GE9	GE Fanuc Series 90-30	SNP	Programming Port	19.2 kbaud, 1 stop bit, 8 data bits, odd parity
	GE Fanuc Series 90-70	SNP	Programming Port	Same as 90-30
-TI3	TI Model 315	CCM	DCU	Auto-Detect
	TI Model 325	CCM	DCU	Auto-Detect
	TI Model 330	CCM	DCU	Auto-Detect
	TI Model 425	CCM	DCM	Auto-Detect
	TI Model 435	CCM	CPU Serial Interface Port	Auto-Detect
-TI5	TI Model 520(C)	TI Direct Connect	CPU Serial Interface Port	Auto-Detect
	TI Model 530(C)	TI Direct Connect	CPU Serial Interface 1Port	Auto-Detect
	TI Model 525	TI Direct Connect	CPU Serial Interface Port	Auto-Detect
	TI Model 535	TI Direct Connect	CPU Serial Interface Port	Auto-Detect
	TI Model 560	TI Direct Connect	CPU Serial Interface Port	Auto-Detect
	TI Model 565	TI Direct Connect	CPU Serial Interface Port	Auto-Detect
	TI Model 545	TI Direct Connect	CPU Serial Interface Port	Auto-Detect
	TI Model 575	TI Direct Connect	CPU Serial Interface Port	Auto-Detect
-PL5	Allen Bradley PLC-5	1785-KE	Computer (Asynchronous) Port	Set Full Duplex & BCC, All others auto-detected
	Allen Bradley PLC-5	1770-KF2	Computer (Asynchronous) Port	Set Full Duplex & BCC, All others auto-detected
-SL5	Allen Bradley PLC-500	1747-KE	Computer (Asynchronous) Port	Set Full Duplex & BCC, All others auto-detected
-OM1	Omron Host Link	Host Link	RS232C Port	Auto-Detect
-PL2	Allen Bradley PLC-2	1771-KA2	Programming Port	Auto-Detect
	Allen Bradley PLC-2	Programming Port	Programming Port	Auto-Detect

MOD Interface Option

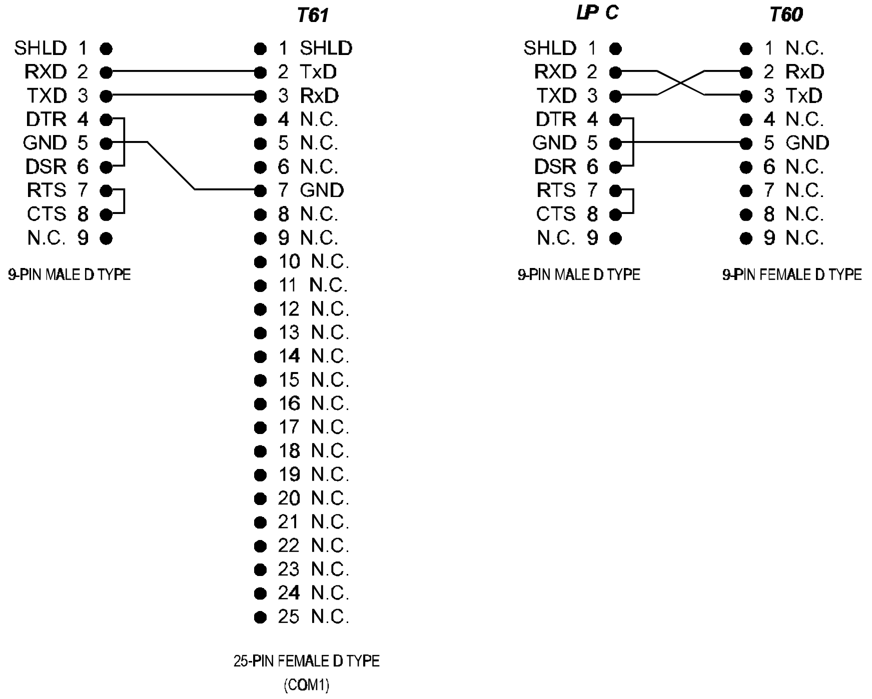
Communications:

The -MOD Option communicates via the Modbus protocol. It was developed to allow the T-60 Series to communicate easily to the Modicon PLC's that use the Modbus protocols (Micro 84, 484, 584, 184/384, 884, 984/381, etc.) It also works well with communication modules available for other PLC's like GE Fanuc's Serial Communications Module (Cat.# IC693CMM311 used in RTU Mode). When used with a Modicon PLC, the communications occurs via the programming port. The T-60 Series does auto-detect for the serial communications parameters, therefore no special procedures are necessary for configuring the serial port on the PLC. The CALL PLCINIT command takes care of configuring COM1 on the T-60 Series device and initializing the communications with the PLC.

Connections:

The -MOD Option comes with the proper cable to interface to the Modicon PLC. For those who need to change the length of the cable, the proper

Figure 22
T-60 Series to Modbus Communication Connections



connections are shown below:

Command Variations:

There are no variations between the descriptions of CALL PLCINIT, CALL PLCREAD, and CALL PLCWRITE in the previous section and the Modbus implementation of those commands.

GE9 Interface Option

Communications:
The -GE9 Option uses GE Fanuc's SNP Protocol to communicate to their Series 90 PLCs. Access to the PLC occurs via the programming port. In the near future, they will also be offering a Serial Communications Module that supports SNP which will allow simultaneous connections to the PLC by more than one T-60 or T-61, as well as a programmer. Check with GE for availability. If you need to allow simultaneous communication to one Series 90 from both a T-60 Series unit and a programming device, GE's Serial Communication Module (catalog # IC693CMM311) supports the Modbus protocol when in RTU mode. You would then need to use the -MOD option to communicate to the PLC. See the previous section for more information. The rest of this section will deal with the -GE9 SNP protocol currently only available through the programming port. (Note: since the original publication of this document, it appears that GE Fanuc has released a version of of Serial Communications Module that does support the SNP protocol.)

Table 8
T60 Series / RS232
Communication Protocol
-GE9 Interface Option

Parameter—	Mode	Baud Rate	Data Bits	Stop Bit(s)	Parity
Setting—	RS422	19.2 kbaud	8	1	Odd

When first sending the CALL PLCINIT command, the T-60 Series unit configures COM1 to have the following parameters:

Make sure that the configuration of the programming port of the PLC matches these parameters, or the CALL PLCINIT command will fail and you will get an error message like "PLC link not established."

Connections:
The -GE9 Option comes with the proper cable to interface to a GE Fanuc Series 90 PLC. For those who need to change the length of the cable, the proper connections are shown below:

Command Variations:
There are no variations between the descriptions of CALL PLCINIT, CALL PLCREAD, and CALL PLCWRITE in the previous section and the GE Series 90 SNP implementation of those commands.

TI3 Interface Option

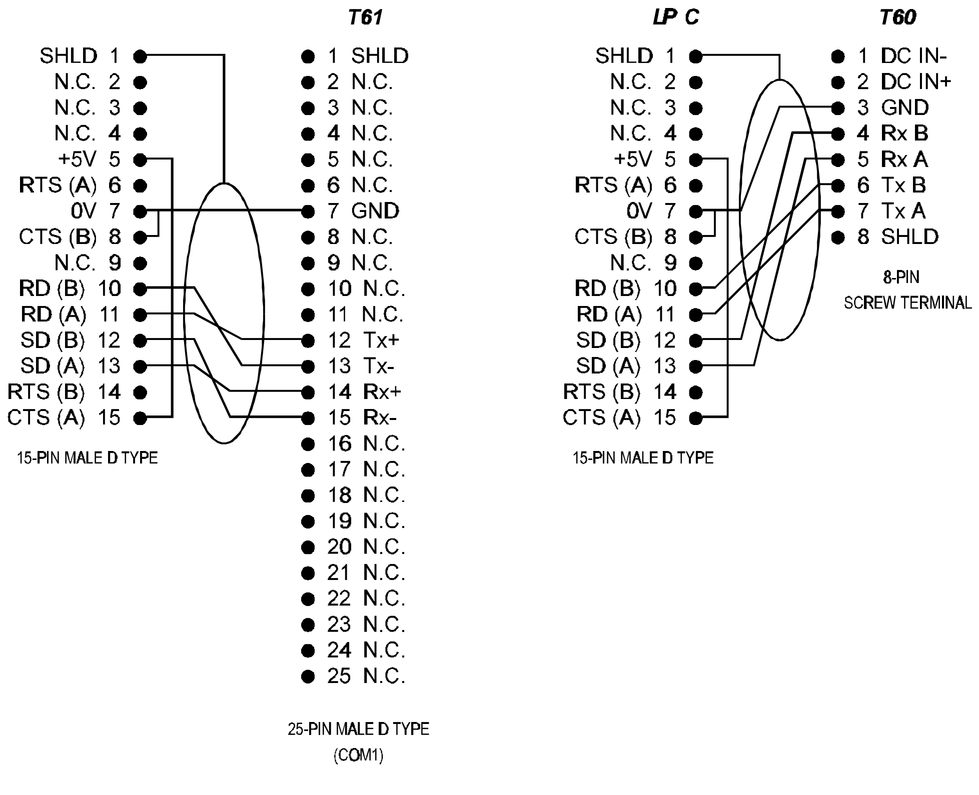
Communications:

The -TI3 Option uses Texas Instruments' CCM protocol for communicating to TI Series 305 and Series 405 PLC's. The Series 305 PLC's (Models 315, 325, and 330) communicate via a DCU module. The communication parameters are auto-detected by the T-60 Series device. Therefore, you don't need to set up any communication parameters in your PLC. The TI Model 425 communicates via the DCM module. It's communication parameters are also auto-detected. The TI Model 435 has a serial communication port built into the CPU module. No other communications module is necessary. As with the other Models, the T-60 Series auto-detects the 435's communication parameters. In all cases, the communication protocol is RS232C, so use the RS232C port, not the RS422 port, if you have a choice.

Connections:

The -TI3 Option comes with the proper cable to interface to a TI 305 or 405 Series PLC. For those who need to change the length of the cable, the proper connections are shown below:

Figure 23
10T-60 Series to GE Series 90 Communication Connections



Command Variations:

TI Series 305 and 405 PLC's return 5 words of network status information when using CALL PLCREAD with cmd = 1. The following example describes a typical way to read the status information:

*Numbers in **bold** cannot be changed*

10 DIM stat%(5)
20 CALL PLCREAD(1,**1**,**0**,**5**,stat%(1))

This will store *Last Error and Previous Error* in stat%(1), *Number of Successful Communications* in stat%(2), *Number of Erroneous Communications* in stat%(3), *Number of Retries For Header* in stat%(4), and *Number of Retries for Data* in stat%(5). (See your PLC manual for more details) **You must read 5 words of data any time you wish to read the network status.**

You can reset the status registers by using the CALL PLCWRITE command with cmd = 1. You must write five words. The data can be arbitrary, since the net result is resetting the status registers, no matter what you write. See the example:

*Numbers in **bold** cannot be changed*

10 DIM newstat%(5)
20 CALL PLCWRITE(1,**1**,**0**,**5**,newstat%(1))

(See your PLC manual for more details) **You must write 5 words of data any time you wish to reset the network status.**

All other commands behave as described in previous sections.

TI5 Interface Option

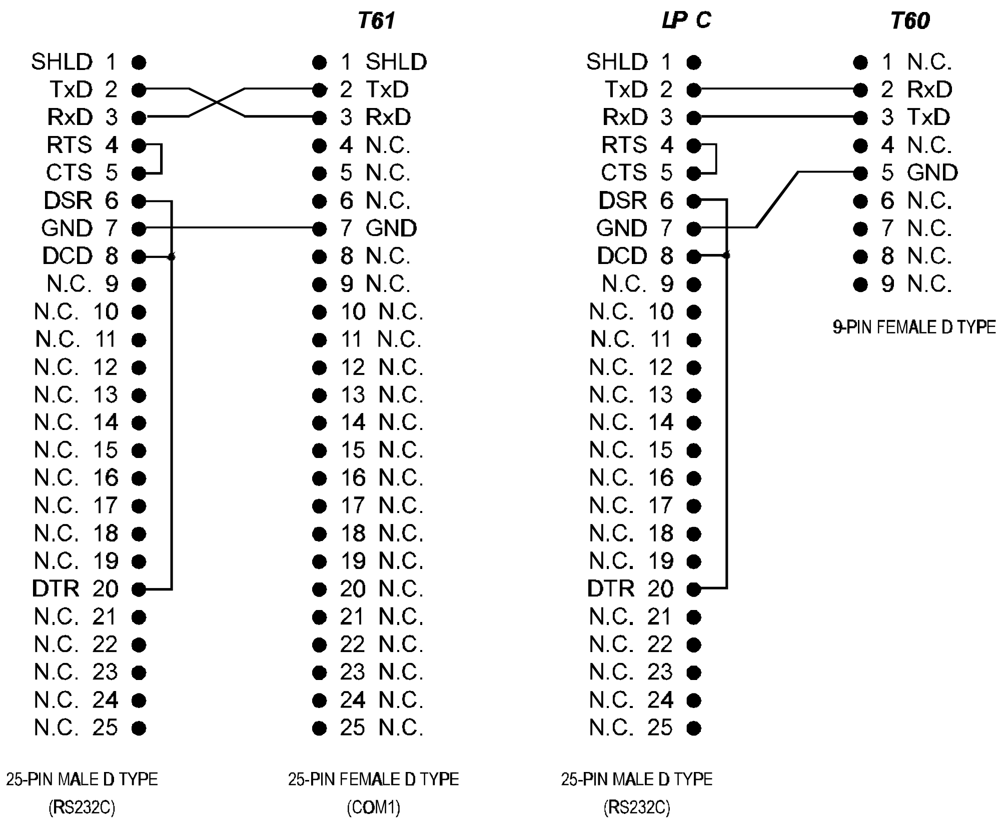
Communications:

The -TI5 Option uses Texas Instruments' Direct Connect protocol for communicating to TI Series 505 PLC's. All models in the TI500/505 Series have an RS232C serial port on the CPU module. The T-60 Series -TI5 Option uses this port for communication to the PLC. Both the T-60 and T-61 auto detect the communication parameters, so no setup is needed to get the two devices communicating. The only caution is to make sure that no passwords exist that would not allow the T-60 Series unit access to the programming port.

Connections:

The -TI5 Option comes with the proper cable to interface to a TI 505 Series PLC. For those who need to change the length of the cable, the proper connections are shown below:

Figure 24
T-60 Series to TI Series 305 or TI Series 405 RS232C Communication Connections



T61
LP C
T60
SHLD 1, TxD 2, RxD 3, RTS 4, CTS 5, DSR 6, GND 7, DCD 8, N.C. 9–19, DTR 20, N.C. 21–25
1 SHLD, 2 TxD, 3 RxD, 4–6 N.C., 7 GND, 8–25 N.C.
1 N.C., 2 RxD, 3 TxD, 4 N.C., 5 GND, 6–9 N.C.
25-PIN MALE D TYPE (RS232C)
25-PIN FEMALE D TYPE (COM1)
25-PIN MALE D TYPE (RS232C)
9-PIN FEMALE D TYPE

Command Variations:

The following table will help make the terminology of the previous sections make more sense with respect to the TI Series 505 documentation. It is a cross reference between the Memory Type in the PLC documentation and the cmd for the CALL PLCREAD and CALL PLCWRITE commands.

Table 9
Cross Reference
-TI5 Memory Type Vs. Call
PLCREAD / PLCWRITE

READ	WRITE	Cmd	Memory Type	Range	Data / Response
•		1	STW Memory	1-15	1 word per register
•		2	X Memory	1-1024	1 word per 16 bits
•	•	3	Y Memory	1-1024	1 word per 16 bits
•	•	4	V Memory	1-NNNN	1 word per register
		5	N/A	N/A	N/A
		6	N/A	N/A	N/A
•		7	WX Memory	1-1024	1 word per register
•	•	8	WY Memory	1-1024	1 word per register
•	•	9	CR Memory	1-32768	1 word per 16 bits

PL5 Interface Option

Communications:
The -PL5 Interface Option allows the T-60 Series to communicate to the Allen Bradley PLC-5 (Data Highway Plus™) through the 1785-KE or 1770-KF2 Series B Data Highway™ RS-232-C Interface Modules. The communications cable is EMERSON EMC's standard Null Modem cable (N-MODEM-25-25 for the T-61 and N-MODEM-9-25 for the T-60).

Connections:
The -PL5 Option comes with the proper cable to interface direct to a PLC-520,540 or 580 DF1 port. You may also choose a cable to interface to either a 1785-KE or a 1770-KF2 Series B Data Highway™ RS-232-C Interface Module. The cable sections are as follows:

Using the -PL5 with a 1785 KE Module:
T-60 - 20-00138-01
T-61 - 20-00141-01

Using the -PL5 with a 1770 KF2 Module:
T-60 - 20-00109-02
T-61 - 20-00109-01

Using the -PL5 with a PLC-520, 540, or 580 DF1 port:
T-60 - 20-00109-02
T-61 - 20-00141-01

Refer to the end of this section for the schematics for the above cables.

Allen-Bradley 1785 KE and 1770 KF2 Set-Up:
Correctly setting the dip switches on the 1785 KE or 1770 KF2 Interface Module is an important step in the set up process. Furthermore it is a good idea to make sure that your 1785 KE or 1770 KF2 is working properly by communicating to the PLC5 via the KE or KF2 interface and Allen Bradley's programming software (APS). Follow the appropriate Allen-Bradley user's manual for setting up your system to communicate in this fashion. Once you have successfully communicated with between the PC and your PLC, communicating with the Eason is a snap.

Table 10
1785 KE Interface Module Dip Switch Settings

	SWITCH NUMBER							
	1	2	3	4	5	6	7	8
SW1	Up	Up	Up	Up	Up	Up		
SW2	Dn	Dn	Up	Up	Up	Up	Up	Up
SW3	Dn	Dn	Up	Dn	Dn	Dn		
SW4	Up	Up						

Configure the 1785 KE dip switches in the following manner:

Configure the 1770 KF2 dip switches in the following manner:

Table 11
1770 KF2 Interface Module Dip Switch Settings

	SWITCH NUMBER					
	1	2	3	4	5	6
SW1	Dn	Dn	Up	Dn	Dn	
SW2	Dn	Dn				
SW3	Up	Up	Up			
SW4	Up	Up	Up			
SW5	Up	Up				
SW6	Dn	Up	Up	Up		
SW7	Up	Dn				
SW8	Dn	Up				

Command Variations:
The CALL PLCREAD and CALL PLCWRITE commands for the PLC-5 are significantly different from the other PLC's described here. The following descriptions apply to the -PL5 Option only:

CALL PLCREAD Statement

Syntax:

CALL PLCREAD*(id, file, address, [bit], count, variable/array* **)**

Comments:

This command is specific to the -PL5 Interface Option.

id specifies the address of the PLC from which you wish to read data. This number is usually 1 when interfacing to one PLC.

file specifies the file number that you wish to access.

address is the address of the first element to access in the above *file.*

[bit] is an optional parameter which specifies the starting bit location of the bits you wish to read. **If you are reading words (as in reading a register value), leave this field blank (ie, ...***address,,count,...).* If you are reading bits (I/O points, internal coils, etc.) specify the starting bit in the element you are addressing (*address)*. This could be any bit from 0-15.

count specifies the number of bits and/or elements that you wish to read. If you are reading data in the form of words, it is the number of consecutive elements you wish to read. If you are reading bits, it is the number of consecutive bits you wish to read.

variable/array is the variable name or single dimension array name where you wish to store the data you are reading. If the value of *count* (see above) is 1, this will be a variable expression. If you are reading elements that are stored as words and *count* is greater than one, this will be an array (make sure you properly dimension the array prior to using it). If you are reading the status of more than one, but less than 16 bits, you will be reading one word, and therefore can use a discrete variable name. If you are reading more than 16 bits, you will need to use an array. The dimension of the array variable will be the next integer greater than the desired number of bits divided by 16. For example, if you wish to read the status of bits 1-24 you will need to dimension your array to at least two since 24/16 = 1.5 and two is the next greater integer. Remember, any time you are reading more than one word of data (more than one register or more than 16 bits) you must use a **dimensioned array variable**.

Examples:

```
10 DIM regdat%(10)
20 CALL PLCREAD(1,4,16,,5,regdat%(4))
```

This command returns the contents of element 16 in regdat%(4), element 17 in regdat%(5), element 18 in regdat%(6), element 19 in regdat%(7), element 20 in regdat%(8), from file #4 in the PLC-5 with id #1.

```
10 CALL PLCREAD(2,5,47,,1,regdat1%)
```

This command reads the data in input element number 47 in file #5 in the PLC-5 with id (address) = 2 and stores it in the variable regdat1%

```
10 DIM stat%(2)

20 CALL PLCREAD(1,1,1,15,18,stat%(1))
```

This command would access file 1 in the PLC with an id of 1. It would return the status of address 1, bit 15, and address 2, bits 0-14 in stat%(1) and the status of address 2, bit 15 and address 3 bit 0 (and 14 zeros) in stat%(2).

CALL PLCWRITE Statement

Syntax:

CALL PLCWRITE(*id, file, address, [bit], count, variable/array* **)**

Comments:

This command is specific to the -PL5 Interface Option.

id specifies the address of the PLC in which you wish to write data. This number is usually 1 when interfacing to one PLC.

file specifies the file number that you wish to write to.

address is the address of the first element to write to in the above *file.*

[bit] is an optional parameter which specifies the starting bit location of the bits you wish to write. **If you are writing words (as in writing a register value), leave this field blank (i.e., ...***address,,count,...).* If you are writing bits (I/O points, internal coils, etc.) specify the starting bit in the element you are addressing (*address)*. This could be any bit from 0-15.

count specifies the number of bits and/or elements that you wish to set. If you are writing data in the form of words, it is the number of consecutive elements you wish to write. If you are setting bits, it is the number of consecutive bits you wish to set.

variable/array is the variable name or single dimension array name where you wish to store the data you are writing. If the value of *count* (see above) is 1, this could be a variable expression. If you are writing elements that are stored as words and *count* is greater than one, this will be an array (make sure you properly dimension the array prior to using it). If you are setting more than one, but less than 16 bits, you will be writing one word, and therefore can use a discrete variable name. If you are setting more than 16 bits, you will need to use an array. The dimension of the array variable will be the next integer greater than the desired number of bits divided by 16. For example, if you wish to set bits 1-24 you will need to dimension your array to at least two since 24/16 = 1.5 and two is the next greater integer. Remember, any time you are writing more than one word of data (more than one register or more than 16 bits) you must use a **dimensioned array variable.**

Examples:

```
10 DIM regdat%(10)
20 CALL PLCWRITE(1,4,16,,5,regdat%(4))
```

This command writes the contents of regdat%(4) in element 16, regdat%(5) in element 17, regdat%(6) in element 18, regdat%(7) in element 19, and regdat%(8) in element 20 to file #4 in the PLC-5 with id (address) of 1.

```
10 CALL PLCWRITE(2,5,47,,1,regdat1%)
```

This command writes the data in regdat1% into element number 47 in file #5 in the PLC-5 with id (address) of 2.

```
10 DIM stat%(2)
20 CALL PLCWRITE(1,2,33,12,5,13)
```

This command sets bits 12, 14, and 15 to a 1, and bits 13 and 16 to a 0 in element 33 in file #2 in PLC with an id (address) of 1.

-SL5 Interface Option

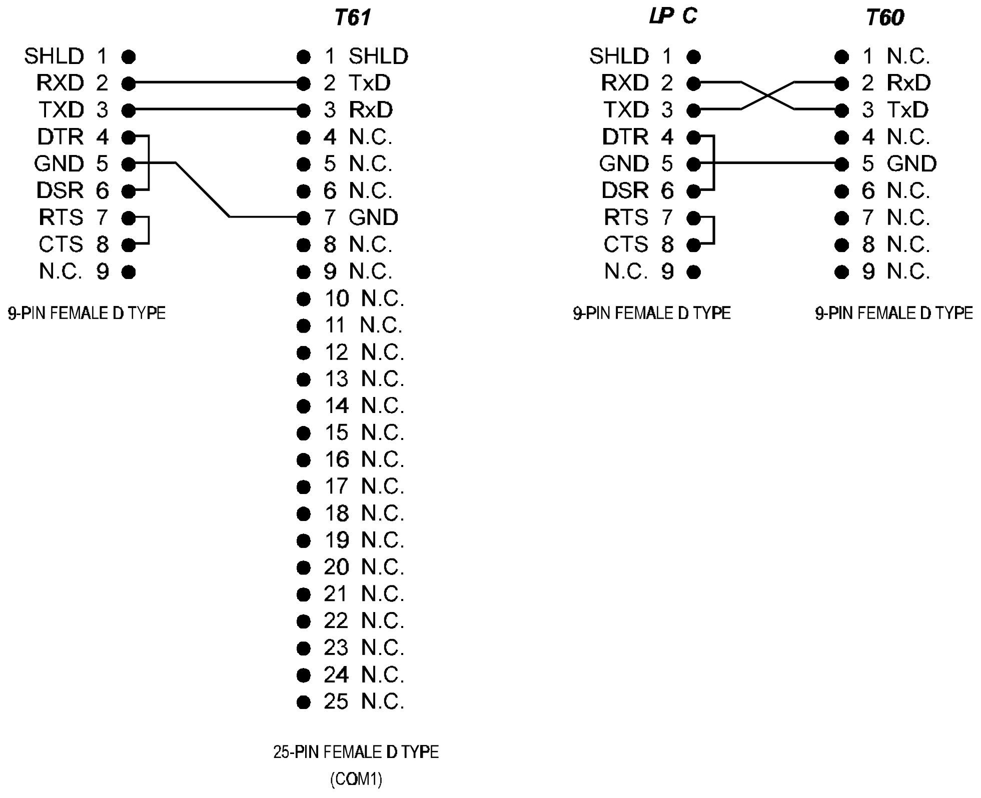
Communications:

The -SL5 Interface Option allows the T-60 Series to communicate to the Allen Bradley SLC-500 (DH-485Ô) through the 1747-KE DH-485Ô to RS-232-C Interface Module. The communications cable is EMERSON EMC's standard Null Modem cable (N-MODEM-25-9 for the T-61 and N-MODEM-9-9 for the T-60).

Connections:

The -SL5 Option comes with the proper cable to interface to a 1747-KE Interface Module. For those who need to change the length of the cable, the proper connections are shown below:

Figure 25
T-60 Series to Allen Bradley 1747-KE Module Communication Connections



SLC-500 Setup:

The SLC-500 needs to have a 1747-KE interface module installed in it. If the processor is a stand-alone type (SLC500), you may have to add a two slot option rack to add this interface module. Some newer SLC-500's have a built-in serial port, this port can be used to gain direct access to SLC-500 by using the DF1 protocol. This document refers only to the setup and use of the 1747-KE module. Please consult EMERSON EMC for other applications.

Follow the setup guidelines for the 1747-KE module exactly. If possible, use your Allen-Bradley programming software to communicate to the SLC-500 once you perform all of the setup operations. In general you can use the default settings as long as you change the DF1 Port Setup Parameters, select the correct node address, and select FULL DUPLEX operation. If you want to check all of the parameters using a terminal connected to the setup port, use the following parameters:

DF1 Port Setup Parameters:

19.2K baud (this is not critical, the -SL5 option will auto-baud and find yourbaud rate)
8 data bits
No parity
1 stop bit.

DH-485 Port Setup Parameters:

Node Address - 2 Set the PLC to node address 1 (this is performed with the Allen-Bradley setup software for the PLC). The -SL5 interface will reside at node address 0.
Max Node Address - 31
Message Time-out - 1000ms
Pass Through - Enabled
Baud Rate - 19200

DF1 Protocol Menu:

Full Duplex

DF1 Protocol Full Duplex Setup Menu:

Duplicate Packet Detection - Disabled
Checksum - BCC
Constant Carrier Detect - Disabled
Modem Init String - (blank)
Embedded Response Detect - Embedded Response
ACK Time-out - 1.0 Seconds
ENQuriy Retries - 2
NAK Received Retries - 2

Be sure to return the 1747-KE module to the Run mode (jumper settings), and jumper the 1747-KE for RS-232. Connect the DF1 port to COM1 on the T-60 Series product, and you should be able to communicate. Try sending a CALL PLCINIT(1,1) to see if you get a Ready response. If you do, start programming! If not, try checking that you are using COM1 on the T-60 Series product. Make sure you are plugged into the DF1 port, not the configuration port on the 1747-KE. Make sure you are no longer in setup mode for the 1747-KE, and that the jumpers are set to RS-232.

Command Variations:
The CALL PLCREAD and CALL PLCWRITE commands for the SLC-500 are significantly different from the other PLC's described here. *The following descriptions apply to the -SL5 Option only:*

CALL PLCREAD Statement

Syntax:

CALL PLCREAD(*id, type, file, address, [bit], count, variable/array* **)**

Comments:

This command is specific to the -SL5 Interface Option.

id specifies the address of the PLC from which you wish to read data. This number is usually 1 when interfacing to one PLC. See the SLC-500 Setup section for more information on the selection of the *id*.

type specifies the file type that is required for a specific file. The allowable file types and their use are as follows:

0	Outputs - The -SL5 option will not allow direct access to I/O.
1	Inputs - The -SL5 option will not allow direct access to I/O.
2	Status - S file types
3	Bit - B file types
4	Timer - T file types
5	Counter - C file types
6	Control - R file types
7	Integer - N file types

file specifies the file number that you wish to access. The use of a specific file is restricted to files which your program access. For instance, if your program uses no timers, and you access a timer file you will get a “PLC LINK NOT ESTABLISHED” error.

address is the address of the first element to access in the specified *file.* Allen-Bradley restricts reading from or writing to locations which are not specified within a program. For example if your program only access N7:0 through N7:4 and you try to read from N7:5 (one address higher than your program access), you will get an error. To avoid this problem, we suggest that your program access data at least one word higher than words that the -SL5 interface is trying to access. When addressing bit files (type 3), you must specify the WORD address, rather than the BIT address. For example to access B3:250 you need to think of accessing word address B3:15/10. See the examples at the end of this section for further clarification.

bit is the starting bit location of the bits you wish to read. **If you are reading words (as in reading a register value), leave this field blank (i.e., ...***address,,count,...).* If you are reading bits in **any** file type, specify the starting bit in the element you are addressing (*address)*. This could be any bit from 0-15.

count specifies the number of bits and/or elements that you wish to read. If you are reading data in the form of words, it is the number of consecutive elements you wish to read. If you are reading bits, it is the number of consecutive bits you wish to read.

variable/array is the variable name or single dimension array name where you wish to store the data you are reading. If the value of *count* (see above) is 1, this will be a variable expression. If you are reading elements that are stored as words and *count* is greater than one, this will be an array (make sure you properly dimension the array prior to using it). If you are reading the status of more than one, but less than 16 bits, you will be reading on word, and therefore can use a discrete variable name. If you are reading more than 16 bits, you will need to use an array. The dimension of the array variable will be the next integer greater than the desired number of bits divided by 16. For example, if you wish to read the status of bits 1-24 you will need to dimension your array to at least two since 24/16 = 1.5 and two is the next greater integer. Remember, any time you are reading more than one word of data (more than one register or more than 16 bits) you must use a **dimensioned array variable**.

Examples:

```
10 DIM regdat%(5)
20 CALL PLCINIT(1,1)
30 CALL PLCREAD(1,7,7,16,,5,regdat%(1))
```

Line 10 dimensions the array regdat%(10) for future use, line 20 initializes the PLC (this only needs to be done once in your program). Line 30 returns the contents of N7:16 in regdat%(1), N7:17 in regdat%(2), N7:18 in regdat%(3), N7:19 in regdat%(4), and N7:20 in regdat%(5) in the SLC-500 with a node address of 1.

```
10 CALL PLCREAD(2,5,5,3,,1,reg%)
```

This command reads the data in C5:3 in an SLC-500 with node address of 2 and stores it in the variable reg%

```
20 CALL PLCREAD(1,3,3,1,8,15,stat%)
```

This command would access B3:1 in the PLC with a node address of 1. It would return the status of B3:1/8 through B3:2/7 stat%. Note that if your PLC's program does not access any elements from B3:2/8 through B3:255/15, an error will result. This is because the SLC-500 protects (disables external access) to elements which are above the highest accessed elements in a file. Normally this is not a problem for most file types, and status types. B type files however, are protected in bytes. The -SL5 interface reads and writes in words. Therefore, if the -SL5 interface accesses any low bits within the PLC (bits 0 through 7), make sure that your PLC program accesses any bits in the next higher byte. The easiest way to insure that you will not have a problem is to make sure your PLC program accesses the next higher word in memory.

CALL PLCWRITE Statement

Syntax:

CALL PLCWRITE(*id,type, file, address, [bit], count, variable/array* **)**

Comments:

This command is specific to the -SL5 Interface Option.

id specifies the address of the PLC in which you wish to write data. This number is usually 1 when interfacing to one PLC. See the SLC-500 Setup section for more information on the selection of the *id*.

type specifies the file type that is required for a specific file. The allowable file types and their use are as follows:

0	Outputs - The -SL5 option will not allow direct access to I/O.
1	Inputs - The -SL5 option will not allow direct access to I/O.
2	Status - S file types
3	Bit - B file types
4	Timer - T file types
5	Counter - C file types
6	Control - R file types
7	Integer - N file types

file specifies the file number that you wish to write to. The use of a specific file is restricted to files which your program access. For instance, if your program uses no timers, and you access a timer file you will get a “PLC LINK NOT ESTABLISHED” error.

address is the address of the first element to write to in the above *file*.. Allen-Bradley restricts reading from or writing to locations which are not specified within a program. For example if your program only access N7:0 through N7:4 and you try to read from N7:5 (one address higher than your program access), you will get an error. To avoid this problem, we suggest that your program access data at least one word higher than words that the -SL5 interface is trying to access. When addressing bit files (type 3), you must specify the WORD address, rather than the BIT address. For example to access B3:250 you need to think of accessing word address B3:15/10. See the examples at the end of this section for further clarification.

bit is the starting bit location of the bits you wish to write. **If you are writing words (as in writing a register value), leave this field blank (i.e., ...***address,,count,...).* If you are writing bits (I/O points, internal coils, etc.) specify the starting bit in the element you are addressing (*address)*. This could be any bit from 0-15.

count specifies the number of bits and/or elements that you wish to set. If you are writing data in the form of words, it is the number of consecutive elements you wish to write. If you are setting bits, it is the number of consecutive bits you wish to set.

variable/array is the variable name or single dimension array name where you wish to store the data you are writing. If the value of *count* (see above) is 1, this could be a variable expression. If you are writing elements that are stored as words and *count* is greater than one, this will be an array (make sure you properly dimension the array prior to using it). If you are setting more than one, but less than 16 bits, you will be writing one word, and therefore can use a discrete variable name. If you are setting more than 16 bits, you will need to use an array. The dimension of the array variable will be the next integer greater than the desired number of bits divided by 16. For example, if you wish to set bits 1-24 you will need to dimension your array to at least two since 24/16 = 1.5 and two is the next greater integer. Remember, any time you are writing more than one word of data (more than one register or more than 16 bits) you must use a **dimensioned array variable.**

Examples:

```
10 DIM regdat%(5)
20 CALL PLCINIT(1,1)
30 CALL PLCWRITE(1,7,7,16,,5,regdat%(1))
```

Line 10 dimensions the array regdat%(10) for future use, line 20 initializes the PLC (this only needs to be done once in your program). Line 30 writes the contents of regdat%(1) in N7:16, regdat%(2) in N7:17, regdat%(3) in N7:18, regdat%(4) in N7:19, and regdat%(5) in N7:20 in the SLC-500 with a node address of 1.

```
10 CALL PLCWRITE(2,5,5,3,,1,reg%)
```

This command writes the data contained in the variable reg% into C5:3 in an SLC-500 with a node address of 2.

```
20 CALL PLCWRITE(1,3,3,1,8,15,stat%)
```

This command would write the contents of variable stat% into the PLC data bits B3:1/8 through B3:2/7. Note that if your PLC's program does not access any elements from B3:2/8 through B3:255/15, an error will result. This is because the SLC-500 protects (disables external access) to elements which are above the highest accessed elements in a file. Normally this is not a problem for most file types, and status types. B type files however, are protected in bytes. The -SL5 interface reads and writes in words. Therefore, if the -SL5 interface accesses any low bits within the PLC (bits 0 through 7), make sure that your PLC program accesses any bits in the next higher byte. The easiest way to insure that you will not have a problem is to make sure your PLC program accesses the next higher word in memory.

```
20 CALL PLCWRITE(1,7,7,15,1,1,0)
```

This command writes a zero to bit location N7:15/1. Note the restrictions mentioned in the example above.

-OM1 Interface Option

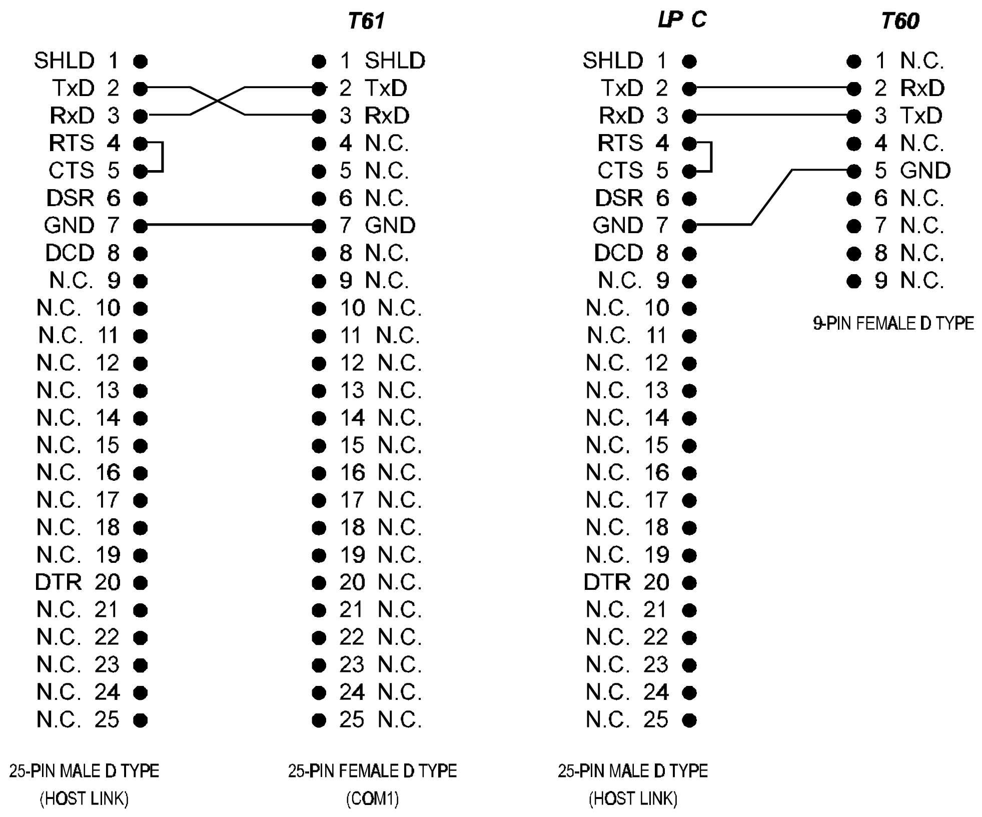
Communications:

The -OM1 Option uses the Omron Host Link protocol for communicating to all suitably equipped Omron PLC's. The Omron Host Link port is connected to the T-60 or T-61's COM1 port via the supplied cable. Both the T-60 and T-61 auto detect the communication parameters, so no setup is needed to get the two devices communicating.

Connections:

The -OM1 Option comes with the proper cable to interface to a Host Link equipped PLC. For those who need to change the length of the cable, the proper connections are shown below:

Figure 26
T-60 Series to Omron Host Link RS232C Communication Connections



Command Variations:

The following table will help the terminology of the previous sections make more sense with respect to the Omron Host Link documentation. It is a cross reference between the Memory Type in the PLC documentation and the *cmd* for the CALL PLCREAD and CALL PLCWRITE commands.

Table 12
Cross Reference
-OM1 Memory Type Vs. Call
PLCREAD / PLCWRITE

READ	WRITE	Cmd	Memory Type	Range	Data / Response
•	•	1	Status Word	*	1 word
•		2	IR & SR	*	1 word per 16 bits
	•	3	IR & SR	*	1 word per 16 bits
•	•	4	DM	*	1 word per register
		5	N/A	*	N/A
		6	N/A	*	N/A
		7	N/A	*	N/A
		8	N/A	*	N/A
•	•	9	LR	*	1 word per 16 bits
•	•	10	HR	*	1 word per 16 bits
•	•	11	AR	*	1 word per 16 bits

PLC Dependent

The -OM1 Option limits you to reading or writing a maximum of 32 words of data (512 bits) during one read or write operation. For example, you are restricted to accessing 32 registers in the DM area with one CALL PLCREAD or CALL PLCWRITE command.

Addressing bits in the -OM1 Option is a little different than in the rest of the PLC interfaces. The -OM1 Option utilizes the Host Link bit addressing scheme which combines the ***WORD*** address and the ***BIT*** address into one number. For the IR, SR, LR, HR, and AR areas data is accessed in this fashion. To read a bit at word 5 bit 3, will require an address of 503, word 11 bit 13 requires the address of 1113. For example, to access the IR data area, word 5, bits 6 through 9, use the following command

CALL PLCREAD(0, 2, 506, 4, A)

This command accesses word 5, bit 6 and reads four bits placing the result in variable A. Note that the four bits will be aligned in variable A with bit 506 in the 2^0 location 507 in the 2^1 location and so on. This makes bit testing within a BASIC program very simple.

Writes to individual bit locations in the Omron PLC is performed via a read modify write process. ***EXTREME*** care should be taken when using data bits which may be updated during a scan. Host Link only allows data to be read and written to on ***WORD*** boundaries. This forces the -OM1 Option to read surrounding bits to obtain an entire word, then sets or clears the desired bits and write the recomposed words back to the PLC. If the data the -OM1 Option reads surrounding the operation is updated during a scan, unpredictable results may occur. Writing to outputs may be something to avoid if your program does not continually update them. Reads do not suffer from this problem.

The Omron status write must be performed in the following manner:

```
CALL PLCWRITE(0,1,0,1,mode)

mode:   0 - program
        1 - debug
        2 - monitor
        3 - run
```

The Omron PLC must be in the monitor mode to enable write commands to perform without errors. A CALL PLCINIT(id, 1) places the PLC into the monitor mode, but if your application has a programming panel or some other peripheral device attached, it may be possible for the operator to disable writes to the PLC. Use the status write function to return the PLC to the monitor mode when necessary.

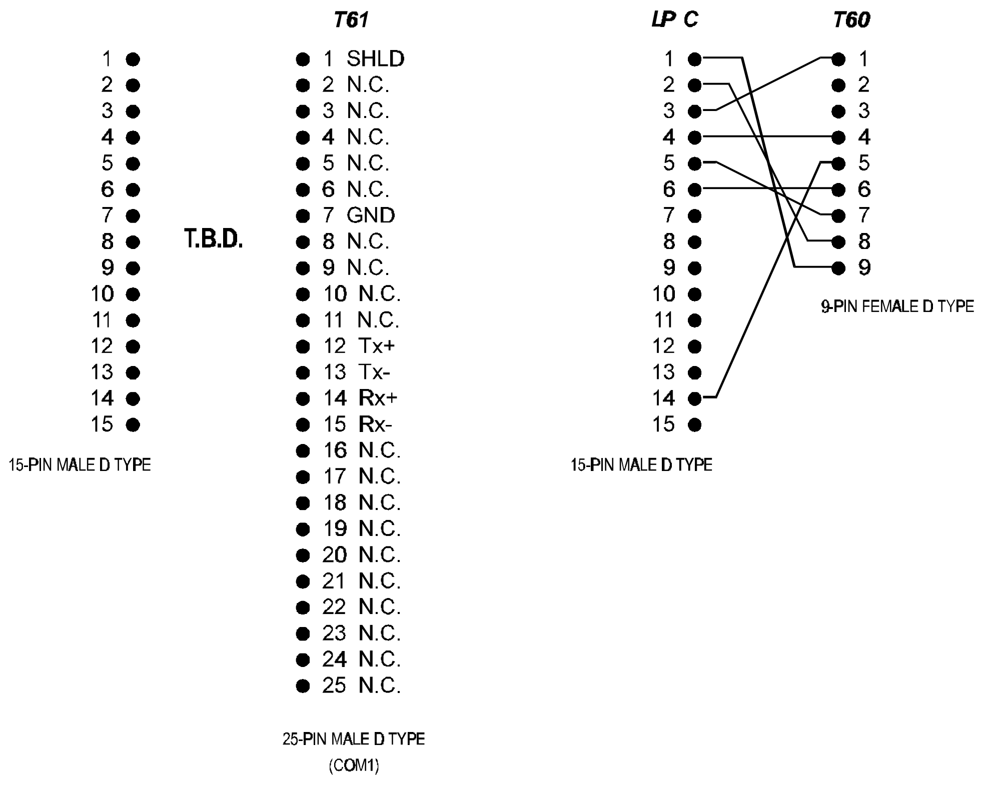
-PL2 Interface Option (Preliminary)

Communications:
The -PL2 Option uses the programming port protocol for communicating to the Allen Bradley PLC-2. The T-60 Series can use the programming port on the PLC-2 CPU or the programming port on the 1771-KA2 Interface Module. The PLC-2 communication parameters are not configurable, so issuing the CALL PLCINIT command will configure COM1 on the T-60 Series device to the proper settings for the PLC-2.

The Data that cannot be accessed in the EMERSON EMC implementation of the PLC-2 protocol as noted in the PLC-2 programming manual (Page C-3 - Data Table Organization) are: Processor Work Areas No. 1 and No. 2, the Reserved Area, the Expanded Data Table and/or User Program Area, and the User Program Area. All other areas are accessible via the CALL PLCREAD and CALL PLCWRITE commands.

Connections:
The -PL2 Option comes with the proper cable to interface to a PLC-2 directly through the programming port, or to the programming port of the 1771-KA2. For those who need to change the length of the cable, the proper connections are shown below:

Figure 27
T-60 Series to Allen Bradley PLC-2 Communication Connections



Command Variations:
There are no variations between the descriptions of CALL PLCINIT, CALL PLCREAD, and CALL PLCWRITE in the previous section and the Allen Bradley PLC-2 or 1771-KA2 implementation of those commands.

-IDEC FA-1J\FA2-J Interface Option

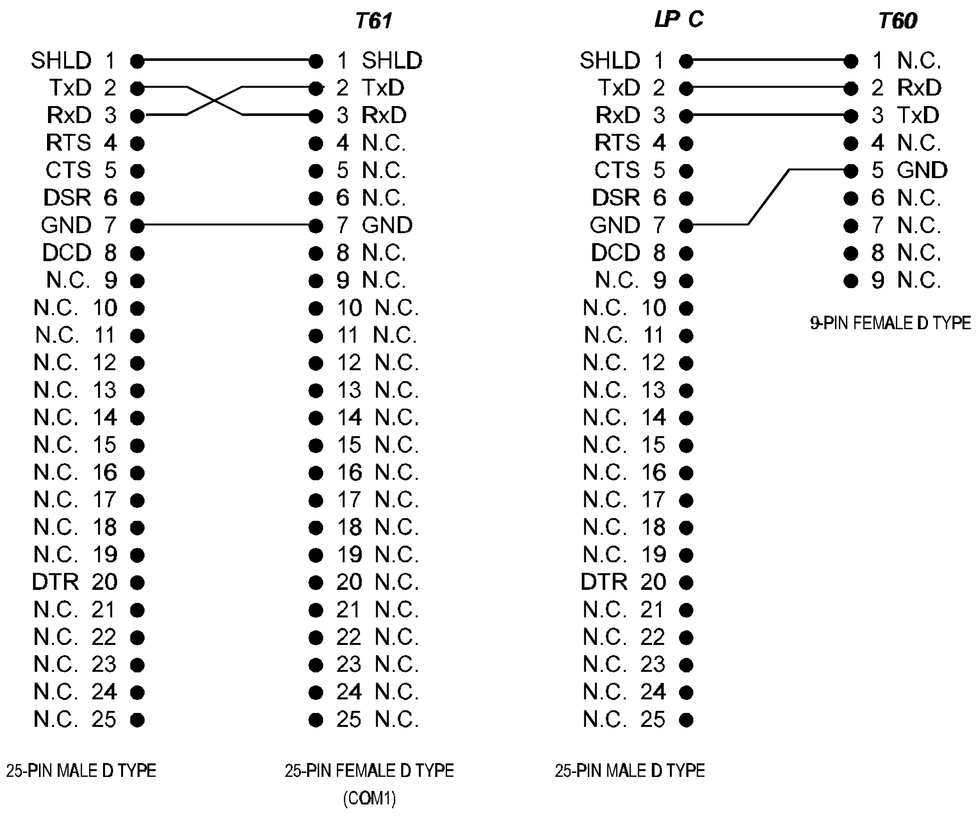
Communications:
The -IDEC option uses the Idec protocol to communicate between the Series T-60 unit and the PLC. The unit is configured by using the *-plcinit(x,y)* command.

Table 13 T60 Series / RS232 Communication Protocol -IDEC FA-1J\FA2-J Interface Option

Parameter—	Mode	Baud Rate	Data Bits	Stop Bit(s)	Parity
FA-1J—	RS232	C9600 baud	8	1	None
FA-2J—	RS232	C9600 baud	8	1	Even

Connections:
In order for the SeriesT-60 and the Idec unit to communicate, a specialized cable is sent with the Eason unit. The pinout for this cable is shown in the table below illustrating the view for both the T-60 and the T-61.

Figure 28 T-60 Series to IDEC FA-1J and FA-2J PLC Communication Connections



Available Commands:

The commands listed in the table below list the commands available for the Idec FA-1J and the Idec FA-2J interfaces.

Table 14
T-60 Series commands and addrss ranges for the Idec FA-1J and the FA-2J

Cmd#	Command	Address	Size	Read	Write	FA-IJ	FA-2J
1	Status	none	BIT	•		•	•
2	In_Out (Input)	0-157	BIT	•		•	•
2	In_Out (Expansion Input)	2000-2157	BIT	•			•
3	In_Out (Output)	200-357	BIT	•	•	•	•
3	In_Out (ExpansionOutput)	2200-2357	BIT	•	•		•
4	Register	800-899	WORD	•	•		•
4	Register (Expansion Register)	1500-1799	WORD	•	•		•
9	Internal Relay	400-717	BIT	•	•	•	•
9	Internal Relay (Expansion Relay)	2400-2717	BIT	•	•		•
12	Timer Value	0-79	WORD	•		•	•
13	Timer Value	0-79	WORD	•	•	•	•
14	Counter Value	0-47	WORD	•		•	•
15	Counter Preset	0-47	WORD	•	•	•	•
16	Shift Register	0-127	BIT	•	•	•	•
17	Ten Mil Timer	T-61-1179	WORD	•		•	•

CALL PLCINIT Statement

Purpose:

This command is used to initialize communication with the IDEC FA-1J and FA-2J series PLC. CALL PLCINIT must be issued prior to any other communication (reading or writing) to the IDEC PLC. CALL PLCINIT setsup specific communications parameters. If the CALL PLCINIT command fails, the T-60 Series unit will break your program and generate the error message “PLC link not established.” If this occurs, make sure that the connections and communications configurations are correct.

Syntax:

CALL PLCINIT(*id,cmd*)

Comments:

id specifies the address of the PLC that you wish to initialize. For the FA-1J and FA-2J series PLC, this number will always be 1.

cmd specifies the type of PLC protocol that you are initiating :

1 - FA-1J on com port 1
2 - FA-2J on com port 1
3 - FA-1J on com port 2
4 - FA-2J on com port 2

Examples:

10 CALL PLCINIT(1,1)
Establishes communication with the Idec FA-1J on communications port 1.

10 CALL PLCINIT(1,4)
Establishes communication with the Idec FA-2J on communications port 4.

CALL PLCREAD Statement

Purpose:

This command is used to read the value(s) in a PLC's registers, the status of bits, or any other accessible memory location within the PLC.

Syntax:

CALL PLCREAD*(id,cmd,start address,# ofregisters/bits,variable/array)*

Comments:

id specifies the address of the IDEC - always use 1.

cmd specifies the read operation you wish to perform. See the table under ***Available Commands*** for the read operation possibilities:

address - describes where in memory the value is to be read from

count - number of bits to be read (when using command of size WORD this will be a 1)

variable/number - this is the variable or number that is to be written in memory in the case of a the variable that will store the value being read

Examples
CALL PLCREAD(1, 9, 400, 13, a%)
Starting at address 400 in the internal relays of the Idec, this command will instruct the PLC to read the first 13 bits and store the result into a%.

CALL PLCREAD(1, 12, 30, 1, t%)
This will capture the monitor value of a timer.

CALL PLCWRITE Statement

Purpose:

This command is used to write value(s) to the IDEC's register(s), memory location(s), or to force one or more output bits in a PLC.

Syntax:

CALL PLCWRITE*(id,cmd,start address,# of registers/bits,expression/variable/array)*

Comments:

id specifies the address of the IDEC **-** always use 1.

cmd specifies the write operation you wish to perform. See the table below for the write operation possibilities:

address - describes where in memory the value is to be stored or read from

count - number of bits to be read (when using command of size WORD this will be a 1)

variable/number - this is the variable or number that is to be written in memory in the case of a write or the variable that will store the value being read

Examples:

CALL PLCWRITE(1, 4, 805, 1, 12345)
This command tells the Idec PLC to write in register #805 the value 12345. This function is only available in the FA-2J PLC.

CALL PLCWRITE(1, 9, 400, 13, 4077)
This instructs the PLC to write into the internal relays at address 400, the value of 4077 using 13 bits.

-Mitsubishi FX PLC Interface (-MFX)

Communications:

The -MFX Option uses the Mitsubishi FX's RS422 port to communicate with the T-60 Series products. For the T-60 this utilizes com port 1 and either port is available for use in the T-61.

Connections:

In order for the SeriesT-60 and the FX unit to communicate, a specialized cable is sent with the Eason unit. The pinout for this cable is shown in the table below for the T-60 and the T-61.

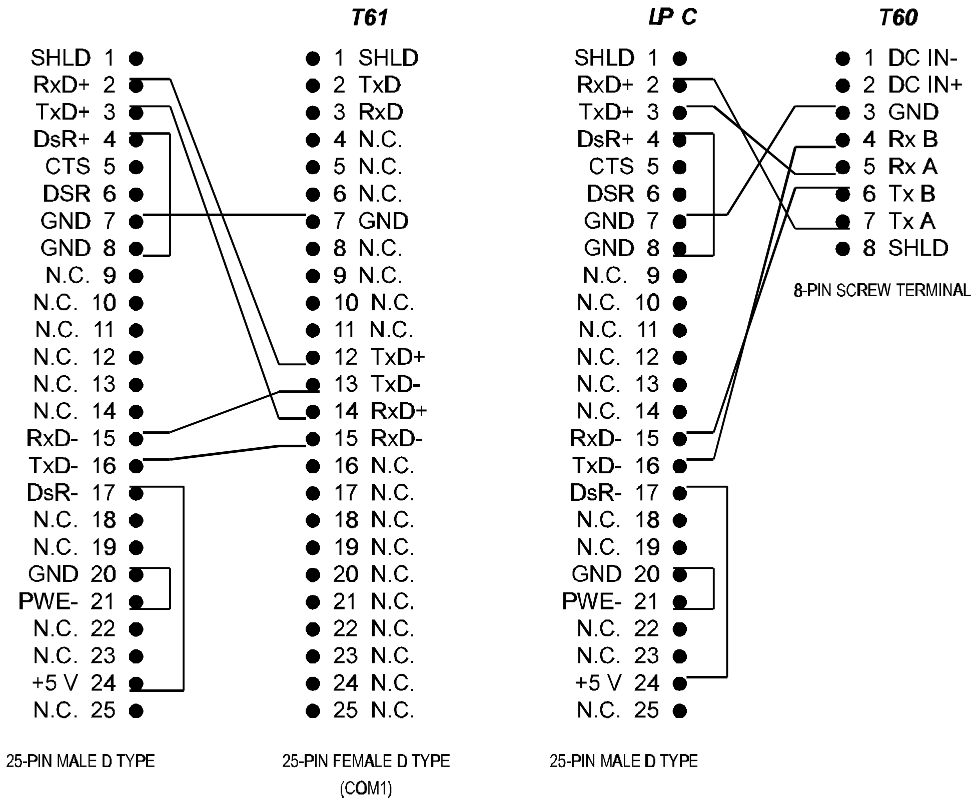


Figure 29
T-60 Series to the Mitsubishi FX series PLC Communication Connections

Available Commands:
The commands listed in the table below list the commands available for the Mitsubishi FX series interfaces.

Table 15
Mitsubishi FX Series Interface Commands

Cmd #	Command	PLC Data Type	Address	Size
1				
2	Inputs	X	0-177	BIT
3	Outputs	Y	0-177	BIT
4	Registers	D	0-511 8000-8255	Word
5				
6				
7				
8				
9	Auxiliary Relays	M	0-1023 8000-8255	BIT
10	Timer Contacts	T	0-255	BIT
11	Counter Contacts	C	0-255	BIT
12	Timer Value	T	0-255	WORD
13				
14	Counter Value	C	0-199	WORD
15				

CALL PLCINIT Statement

Purpose:

This command is used to initialize communication with the FX. CALL PLCINIT must be issued prior to any other communication (reading or writing) to the specified PLC. CALL PLCINIT initializes the BAUD rate to 9600 Baud, 7 data bits, even parity. Either COM 1, COM 2, RS422, and RS232 can be specified. Note that on T-60 's COM1, RS422 is recommended. If the CALL PLCINIT command fails, the T-60 Series unit will break your program and generate the error message “PLC link not established.” If this make sure that the connections and communications configurations are correct. If they are OK, check to make sure that the FX is powered up and ready to accept Commands.

Syntax:

CALL PLCINIT(*id,cmd*)

Comments:

id specifies the address of the FX - always use 1.
cmd specifies the communications mode:

1 - RS422, COM1 - most common and recommended.
2 - RS422, COM2 - T-61 only
3 - RS232 COM 1 - Must use an RS232 to RS422 adapter
4 - RS232 COM 2 - Must use an RS232 to RS422 adapter

Examples:

10 CALL PLCINIT(1,1)
Establish communications with the PLC via the T-60 Series COM 1 RS422 Port.

CALL PLCREAD Statement

Purpose:

This command is used to read the value(s) in a PLC's registers, the status of bits, or any other accessible memory location within the PLC.

Syntax:

CALL PLCREAD(*id,cmd,start address,# of registers/bits,variable/array*)

Comments:

id specifies the address of the FX - always use 1.

cmd specifies the read operation you wish to perform. See the table below for the read operation possibilities:

start address is the starting address of the bit(s) or register(s) you are interested in reading. Inputs and Outputs (data types X and Y) are specified in OCTAL just like you would specify in the PLC ladder logic program. The type specified for this parameter if a variable is to be used is an integer: %.

# of registers/bits is the number of consecutive registers, memory locations, or bits that you wish to read. If you wish to read one bit or register, set this parameter to 1. If you wish to read more than one register or memory location at a time, this number will be the number of consecutive registers or locations you wish to read. If you are reading the status of more than one I/O bit, this number will be the number of consecutive bits you wish to read.

variable/array is the variable name or array name where you wish to store the data you are reading. If you are reading only one register or bit, you may use any variable type you wish. If you are reading multiple registers or more than 16 bits of data, this variable **MUST** be a short integer (%) array variable. If you are reading registers or memory locations that are stored as words and the *# of registers/bits* is greater than one, this will be a short integer array (make sure you properly dimension the array prior to using it). If you are reading the status of more than one, but less than 16 I/O bits, you will be reading one word, and therefore will need to use a single short integer. If you are reading more than 16 bits, you will need to use a short integer array. The dimension of the array variable will be the next integer greater than the desired number of bits divided by 16. For example, if you wish to read the status of bits 1-24 you will need to dimension your array to at least two since 24/16 = 1.5 and two is the next greater integer. Remember, any time you are reading more than one word of data (more than one register or more than 16 bits) you must use a **dimensioned short integer array.**

Examples:

```
10 DIM regdat%(10)
20 CALL PLCREAD(1,4,16,5,regdat%(4))
```

This command returns the contents of register 16 in regdat%(4), register 17 in regdat%(5), register 18 in regdat%(6), register 19 in regdat%(7), register 20 in regdat%(8), from any of the PLC's currently implemented.

```
10 CALL PLCREAD(1,5,47,1,regdat1%)
```

This command reads the data in input register number 47 with id (address) =1 and stores it in the variable regdat1%

CALL PLCWRITE Statement

Purpose:

This command is used to write value(s) to the FX's register(s), memory location(s), or to force one or more output bits in a PLC.

Syntax:

CALL PLCWRITE(*id,cmd,start address,# of registers/bits,expression/variable/array* **)**

Comments:

id specifies the address of the FX - always use 1.

cmd specifies the write operation you wish to perform. See the table below for the write operation possibilities:

start address is the starting address of the bit(s) or register(s) you are interested in writing. Inputs and Outputs (data types X and Y) are specified in OCTAL just like you would specify in the PLC ladder logic program. The type specified for this parameter if a variable is to be used is an integer: %.

# of registers/bits is the number of consecutive registers, memory locations, or bits that you wish to write. If you wish to write one bit or register, set this parameter to 1. If you wish to write more than one register or memory location at a time, this number will be the number of consecutive registers or locations you wish to write. If you are writing the status of more than one I/O bit, this number will be the number of consecutive bits you wish to write.

expression/variable/array is the expression, variable or array data you wish to write to the PLC's I/O, registers, or other memory locations. If the value of *# of registers/bits* (see above) is 1, this will be an expression or a variable. If the number of words is greater than one, this must be an array (make sure you properly dimension the array prior to using it). If you wish to write the status of up to 16 I/O bits, you will be writing one word, and therefore will need to use an expression or discrete variable name. If you wish to set more than 16 consecutive bits, you will need to use an array. The dimension of the array variable will be the next integer greater than the desired number of bits divided by 16. For example, if you wish to set the status of bits 1-24 you will need to dimension your array to at least two since 24/16 = 1.5 and two is the next greater integer. Remember, any time you are writing more than one word of data (more than one register or more than 16bits) you must use a **dimensioned array variable.**

Examples:

```
10 CALL PLCWRITE(1,3,9,2,3)
```
This command writes a 1 to outputs 0 and 1. Note that 3 represents the binary “11" which is the bit pattern desired.

```
10 DIM newdat%(10)
20 CALL PLCWRITE(1,4,5,2,newdat%(4))
```

This command writes the value of newdat%(4) to register 5 and newdat%(5) to register 6 in a PLC with an id (address) = 1.

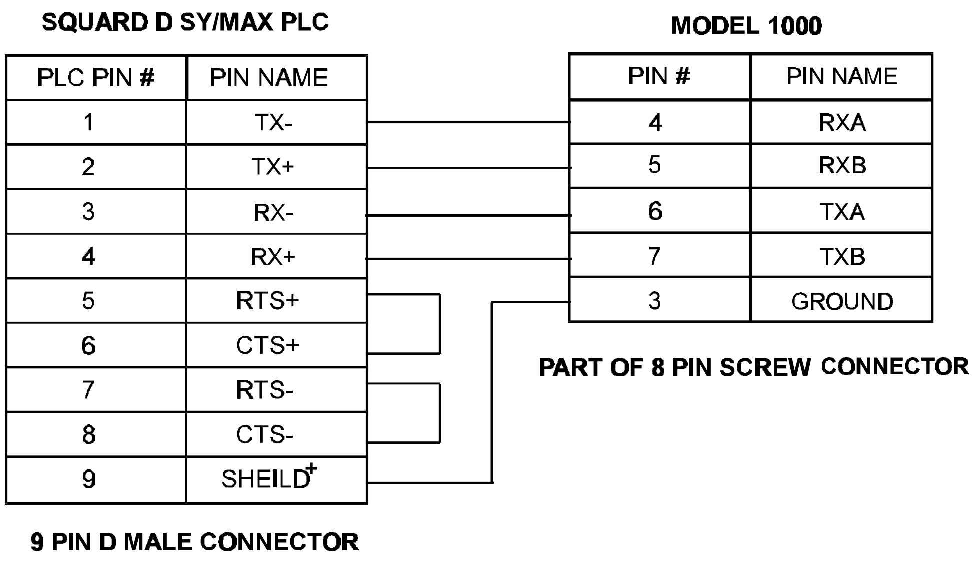
Table 16
PLC/Model/Signal Name

PLC	T-60	Signal Name
7	3	GND
16	4	TXD-
3	5	TXD+
15	6	RXD-
2	7	RXD+
N/C	8	SHIELD
4		DSR+
8		GND
20		GND
21		PWE
17		DSR-

-SQD SQUARE D SY/MAX PLC Interface

Communications:
The SQD PLC option uses the SY/MAX RS422 port to communicate with the T-60 Series products. The connections are as follows:

Figure 30
T-60 Interface Cable:



SQUARD D SY/MAX PLC

PLC PIN #	PIN NAME
1	TX-
2	TX+
3	RX-
4	RX+
5	RTS+
6	CTS+
7	RTS-
8	CTS-
9	SHEILD$^+$

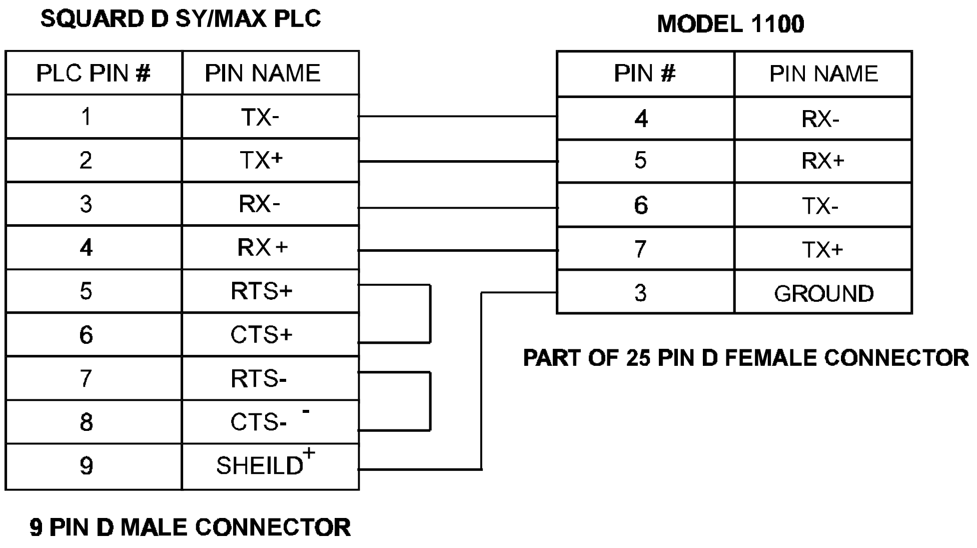
9 PIN D MALE CONNECTOR

MODEL 1000

PIN #	PIN NAME
4	RXA
5	RXB
6	TXA
7	TXB
3	GROUND

PART OF 8 PIN SCREW CONNECTOR

Figure 31
T-61 Interface Cable:



SQUARD D SY/MAX PLC

PLC PIN #	PIN NAME
1	TX-
2	TX+
3	RX-
4	RX+
5	RTS+
6	CTS+
7	RTS-
8	CTS- -
9	SHEILD$^+$

9 PIN D MALE CONNECTOR

MODEL 1100

PIN #	PIN NAME
4	RX-
5	RX+
6	TX-
7	TX+
3	GROUND

PART OF 25 PIN D FEMALE CONNECTOR

CALL PLCINIT Statement

Purpose:

This command is used to initialize communication with the SY/MAX. CALL PLCINIT must be issued prior to any other communication (reading or writing) to the specified PLC. CALL PLCINIT auto detects the baud rate and parity. Either COM 1 or COM 2 can be specified. Note that on T-60 's only COM 1 is available (T-60 's only have one RS422 port). A single non-networked route is specified by the id, see comments below. If the CALL PLCINIT command fails, the T-60 Series unit will break your program and generate the error message "PLC link not established." If this happens, make sure that the connections and communications configurations are correct. If they are OK, check to make sure that the SY/MAX is powered up and ready to accept commands.

Syntax:

CALL PLCINIT(*id,cmd*)

Comments:

id specifies the route for the SY/MAX. Specifying an id of 0 will specify an route of 0,100, an id of 1 will specify a route of 1,101. Networking route specifications will be available in future versions of this interface, check with EMERSON EMC for more details.

cmd specifies the communications mode:

1 - RS422, COM 1
2 - RS422, COM 2 - T-61 only

Examples:

10 CALL PLCINIT(0,1)
Establish communications with the PLC via the T-60 Series COM 1 RS422 Port, specifying a route of 0,100.

CALL PLCREAD Statement

Purpose:

This command is used to read the value(s) in the SY/MAX PLC's registers. It can read words or bits in all allowable addresses for a specific SY/MAX PLC.

Syntax:

CALL PLCREAD(*id,start address,[bit position],# of registers/bits,variable/array*)

Comments:

id specifies the route. See CALL PLCINIT above.

start address is the starting address of the register(s) you are interested in reading.

[bit position] is an optional parameter which specifies the position within a 16 bit register to start reading from. An allowable range for *bit position* is 1 through 16. By specifying this parameter, a bit (single or multiple) read will be performed. For example specifying a *bit position* of 2 will allow the read to place the data at the PLC's bit 2 (2^1) in the bit 0 (2^0 bit) position of the variable/array (Note that all Eason documentation references bits 0 through 15 for bits inside variables in the EASON, while SY/MAX documentation references bits 1 through 16 for registers in the PLC... sorry about the confusion).. Note that only the contents of 1 register's bits may be read at a time. This means that a *bit position* of 9 (2^8 bit) will **only** allow 8 bits to be read. This is due to the fact that there are only 16 bits available in one register, and we are starting at bit 8 this leaves a result of 8 bits.

Omitting the *bit position* parameter will specify that the read will return whole registers rather than bits. Refer to the following examples for samples of how to specify bits or whole registers.

# of registers/bits is the number of consecutive registers, memory locations, or bits that you wish to read. If the bit position parameter is omitted, *# of registers/bits* will specify the number of 16 bit registers to read. If you wish to read more than one register or memory location at a time, this number will be the number of consecutive registers or locations you wish to read. If you have included a bit position parameter, this number will be the number of consecutive bits you wish to read.

variable/array is the variable name or array name where you wish to store the data you are reading. If you are reading only one register or bit, you may use any variable type you wish. If you are reading multiple registers, this variable **MUST** be a short integer (%) array variable. If you are reading registers or memory locations that are stored as words and the *# of registers/bits* is greater than one, this will be a short integer array (make sure you properly dimension the array prior to using it). If you are reading the status of more than one, but less than 16 I/O bits, you will be reading one word, and therefore will need to use a single short integer. Remember, any time you are reading more than one register you must use a **dimensioned short integer array.**

Examples:

```
10 DIM regdat%(10)
20 CALL PLCREAD(1,16,,5,regdat%(4))
```

This command returns the contents of register 16 in regdat%(4), register 17 in regdat%(5), register 18 in regdat%(6), register 19 in regdat%(7), and register 20 in regdat%(8).

```
10 CALL PLCREAD(1,47,,1,regdat1%)
```

This command reads the data in register number 47 with and stores it in the variable regdat1%.

```
10 CALL PLCREAD(1,100,5,2,bits%)
```

This command reads two bits from register 100 starting at bit position 5. The result is placed in the variable bits% with register 100, bit 5 in bit position 0, and register 100 bit 6 in bit position 6.

CALL PLCWRITE Statement

Purpose:

This command is used to write 16 bit words and bits into the SY/MAX PLC's ‘register(s).

Syntax:

CALL PLCWRITE(*id,start address,[bit position], # of registers/bits,expression/variable/array*)

Comments:

id specifies the route. See CALL PLCINIT above.

start address is the starting address of the register(s) you are interested in writing.

[bit position] is an optional parameter which specifies the position within a 16 bit register to start writing to. An allowable range for *bit position* is 1 through 16. By specifying this parameter, a bit (single or multiple) write will be performed. For example specifying a *bit position* of 2 will allow the write to place data contained in the *variable/array* bit 0 (2^0 bit) in the bit 2 (2^1 bit) position of the result (Note that all Eason documentation references bits 0 through 15 for bits inside variables in the EASON, while SY/MAX documentation references bits 1 through 16 for registers in the PLC... sorry about the confusion). Note that only the contents of 1 register's bits may be written at a time. This means that a *bit position* of 9 (2^8 bit) will **only** allow 8 bits to be written. This is due to the fact that there are only 16 bits available in one register, and we are starting at bit 8 this leaves a result of 8 bits.

Omitting the *bit position* parameter will specify that the write will return whole registers rather than bits. Refer to the following examples for samples of how to specify bits or whole registers

# of registers/bits is the number of consecutive registers, memory locations, or bits that you wish to write. If the bit position parameter is omitted, *# of registers/bits* will specify the number of 16 bit registers to write. If you wish to write more than one register or memory location at a time, this number will be the number of consecutive registers or locations you wish to write (up to a maximum of 16). If you have included a bit position parameter, this number will be the number of consecutive bits you wish to write.

variable/array is the variable name or array name where you wish to store the data you are writing. If you are writing only one register or bit, you may use any variable type you wish. If you are writing multiple registers, this variable can either be a short integer (%) array variable or a constant (like 1234). If you are writing registers or memory locations that are stored as words and the *# of registers/bits* is greater than one, this will be a short integer array (make sure you properly dimension the array prior to using it). If you are writing the status of more than one, but less than 16 I/O bits, you will be writing one word, and therefore will need to use a single short integer.

Remember, any time you are writing more than one register you must use a dimensioned short integer array.

Examples:

```
10 CALL PLCWRITE(1,15,1,2,3)
```

This command writes a 1 to register 15 bits 1 and 2. Note that 3 represents the binary “11" which is the bit pattern desired.

```
10 DIM newdat%(10)
20 CALL PLCWRITE(1,5,,2,newdat%(4))
```

This command writes the value of newdat%(4) to register 5 and newdat%(5) to register 6 in a PLC with an id (address) = 1.

```
10 CALL PLCWRITE(1,100,,1,1234)
```

Note that you cannot write to multiple locations with the same data (i.e. CALL PLCWRITE(1,100,,10,1234) - this will generate a “VARIABLE REQUIRED” BASIC error). This operation must be performed by initializing elements of a dimensioned array and writing the array in the following fashion:

```
10 DIM A%(16)
20 FOR N=1 TO 16: A%(N)=1234: NEXT
30 CALL PLCWRITE(1,100,,10,
```

This command writes 1234 to register 100.

Appendix - A
List of Figures

Appendix - B
List of Tables