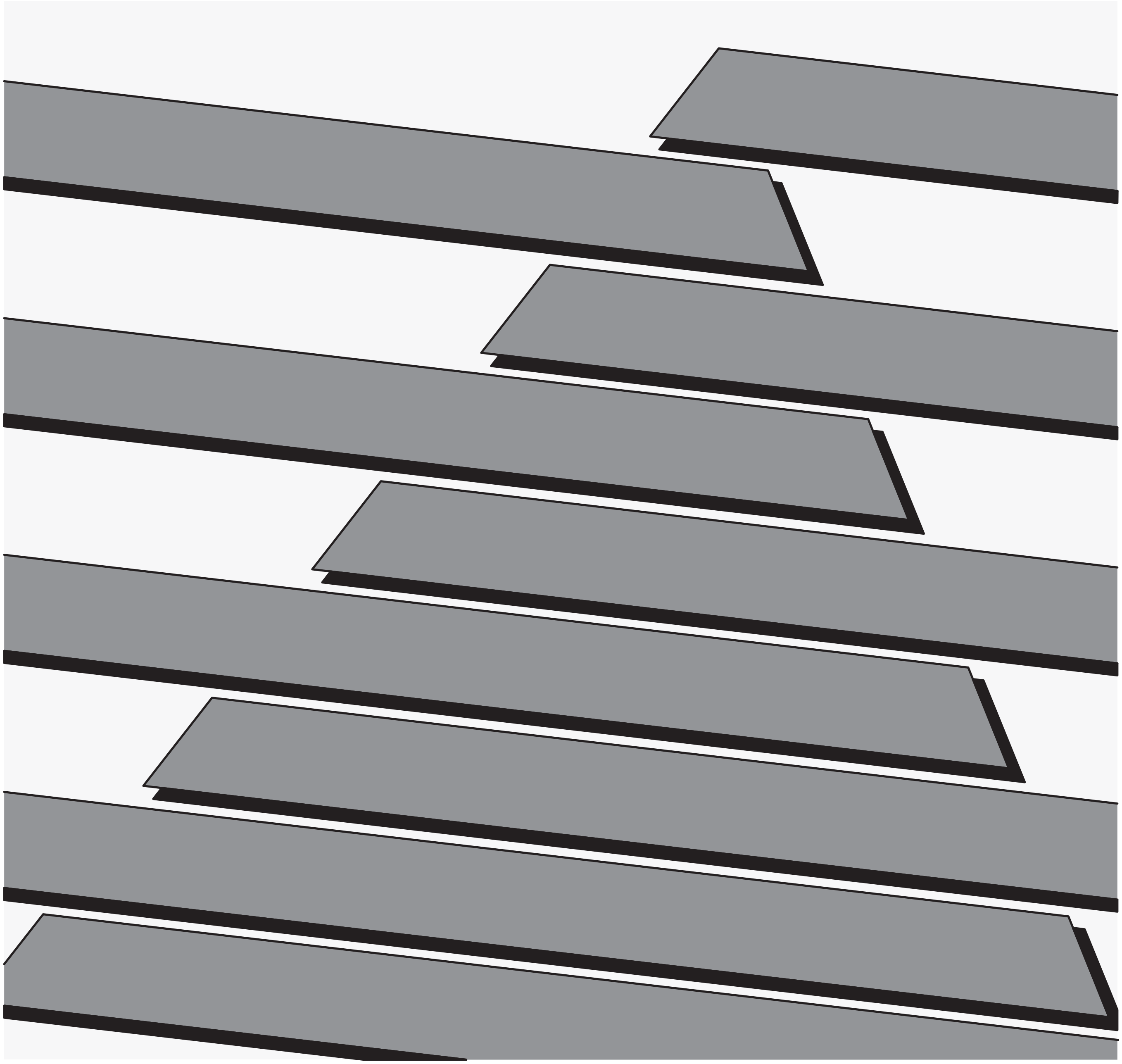
ALLEN–BRADLEY

The Getting Started Guide for HHT

(Catalog Number 1747–PTA1E & 1747–PT1)

User Manual



Important User Information

Solid state equipment has operational characteristics differing from those of electromechanical equipment. “Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls” (Publication SGI-1.1) describes some important differences between solid state equipment and hard–wired electromechanical devices. Because of this difference, and also because of the wide variety of uses for solid state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.

In no event will the Allen-Bradley Company be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, the Allen-Bradley Company cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Allen-Bradley Company with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of the Allen-Bradley Company is prohibited.

Throughout this manual we use notes to make you aware of safety considerations.

ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss.

Attentions help you:

- identify a hazard
- avoid the hazard
- recognize the consequences

Important: Identifies information that is especially important for successful application and understanding of the product.

Preface

Read this preface to familiarize yourself with the rest of the manual. This preface covers the following topics:

- who should use this manual
- the purpose of this manual
- how to use this manual
- conventions used in this manual
- Allen–Bradley support

Who Should Use this Manual

Use this manual if you are responsible for designing, installing, programming, or troubleshooting control systems that use Allen–Bradley small logic controllers.

You should have a basic understanding of SLC 500 products. You should understand programmable controllers and be able to interpret the ladder logic instructions required to control your application. If you do not, contact your local Allen–Bradley representative for information on available training courses before using this product.

Purpose of this Manual

This manual is a learning and reference guide for the HHT. It is an introductory document, designed to allow you to begin programming in the shortest time possible. It does this by focusing on a simple controller and a simple program. Basic concepts are presented, but only with enough detail to get you started and let you know that there is more to be learned.

Contents of this Manual

Chapter:	Title:	Purpose:
1	Setting Up Your Equipment	Shows you how to set up a controller, install your Memory Pak, Battery, and communication cable, and connect your HHT to the controller.
2	Control Basics	Presents basic information you will need to know before you can begin programming with the HHT.
3	Creating a Program	Shows you how to create a program.
4	Online Operations	Shows you how to download (restore) your program to the controller, monitor, and test the program.
Appendix A	Additional Ladder Program Exercises	Introduces you to branching of instructions and the timer instruction.
Appendix B	Troubleshooting Errors	Provides a listing of error messages that you may encounter while working through the guide. Also, offers possible solutions for these errors.
Appendix C	Identifying HHT Function Keys & Instruction Mnemonics	Provides a listing of HHT function keys and their meaning. Also provided is a list of instruction mnemonics.
Glossary	Glossary	Provides definitions of terms used in this guide.

Related Documentation

The following documents contain additional information concerning Allen–Bradley SLC and PLC products. To obtain a copy, contact your local Allen–Bradley office or distributor.

For	Read this Document	Document Number
An overview of the SLC 500 family of products	SLC 500 System Overview	1747–2.30
A description on how to install and use your *Modular* SLC 500 programmable controller	Installation & Operation Manual for Modular Hardware Style Programmable Controllers	1747–804
A description on how to install and use your *Fixed* SLC 500 programmable controller	Installation & Operation Manual for Fixed Hardware Style Programmable Controllers	1747–800
A procedural and reference manual for technical personnel who use an HHT to develop control applications	Allen–Bradley Hand–Held Terminal User's Manual	1747–809
A complete listing of current Automation Group documentation, including ordering instructions. Also indicates whether the documents are available on CD–ROM or in multi–languages.	Automation Group Publication Index	SD499
A glossary of industrial automation terms and abbreviations	Allen–Bradley Industrial Automation Glossary	ICCG–7.1

How to Use this Manual

To use this manual effectively:

- Work through the chapters in sequential order, completing each one before moving on to the next.
- Perform the exercises in appendix A to apply what you have learned in the chapters.
- Consult appendix B to correct and identify any errors you encounter while working through this manual.
- Refer to the glossary for definitions of unfamiliar terms.
- Use the index to locate further information on topics.

Common Techniques Used in this Manual

The following conventions are used throughout this manual:

- Bulleted lists such as this one provide information, not procedural steps.
- Numbered lists provide sequential steps or hierarchical information.
- *Italic* type is used for emphasis.
- Text in `this font` indicates words or phrases you should type.
- Key names match the names shown and appear in bold, capital letters within brackets (for example, **[ENTER]**).

Allen–Bradley Support

Allen–Bradley offers support services worldwide, with 78 Sales Support offices, 494 Authorized Distributors and 242 authorized Systems Integrators located throughout the United States, plus Allen–Bradley representatives in every major country in the world.

Local Product Support

Contact your local Allen–Bradley representative for:

- Sales and Order Support
- Product Technical Training
- Warranty Support
- Support Service Agreements

Technical Product Assistance

If you need to contact Allen–Bradley for technical assistance, please review the information in appendix B, Troubleshooting Errors, first. Then call your local Allen–Bradley representative.

Your Questions or Comments on this Manual

If you have any suggestions for how this manual could be made more useful to you, please send us your ideas on the enclosed reply card.

If you find a problem with this manual, please notify us of it on the enclosed Publication Problem Report.

Setting Up Your Equipment

Chapter 1

Control Basics

Chapter 2

Creating a Program

Chapter 3

Chapter 1

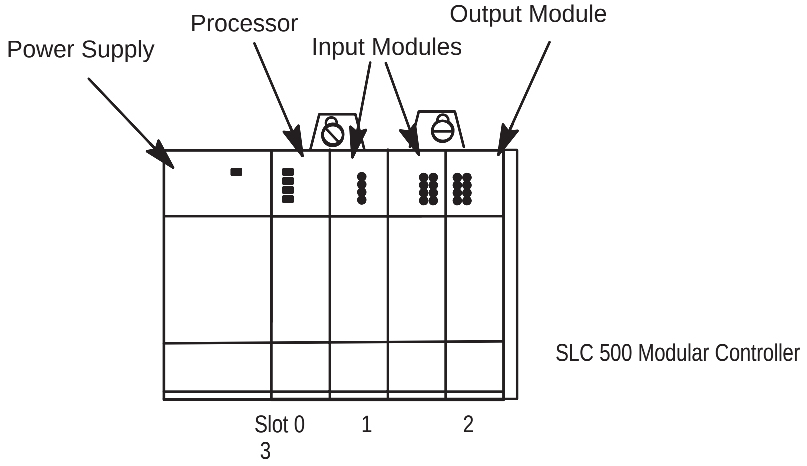
Setting Up Your Equipment

This chapter briefly describes SLC 500 controller styles, then shows you how to set up your equipment in preparation for the exercises in later chapters. Topics include:

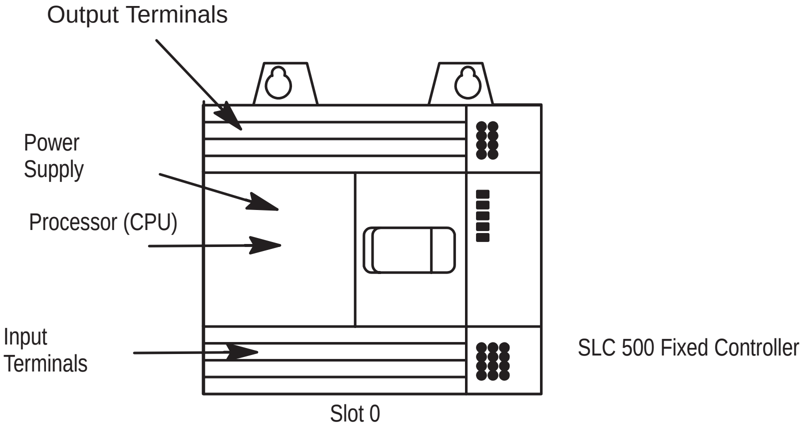
- Controller Styles
- Setting up a Demo Unit
- Setting up a Field–Wired Controller
- Installing the Memory Pak, Battery, and Communication Cable
- HHT Features
- HHT Powerup
- HHT Display Format
- The Keyboard

Controller Styles

The SLC 500 comes in two different styles: modular and fixed. These styles are shown below. The modular controller consists of a rack, power supply, processor (CPU), and Input/Output (I/O) modules. The fixed controller consists of a power supply, processor (CPU), and a fixed number of I/O contained in a single unit. You can add an expansion rack to the fixed controller.



SLC 500 Modular Controller



SLC 500 Fixed Controller

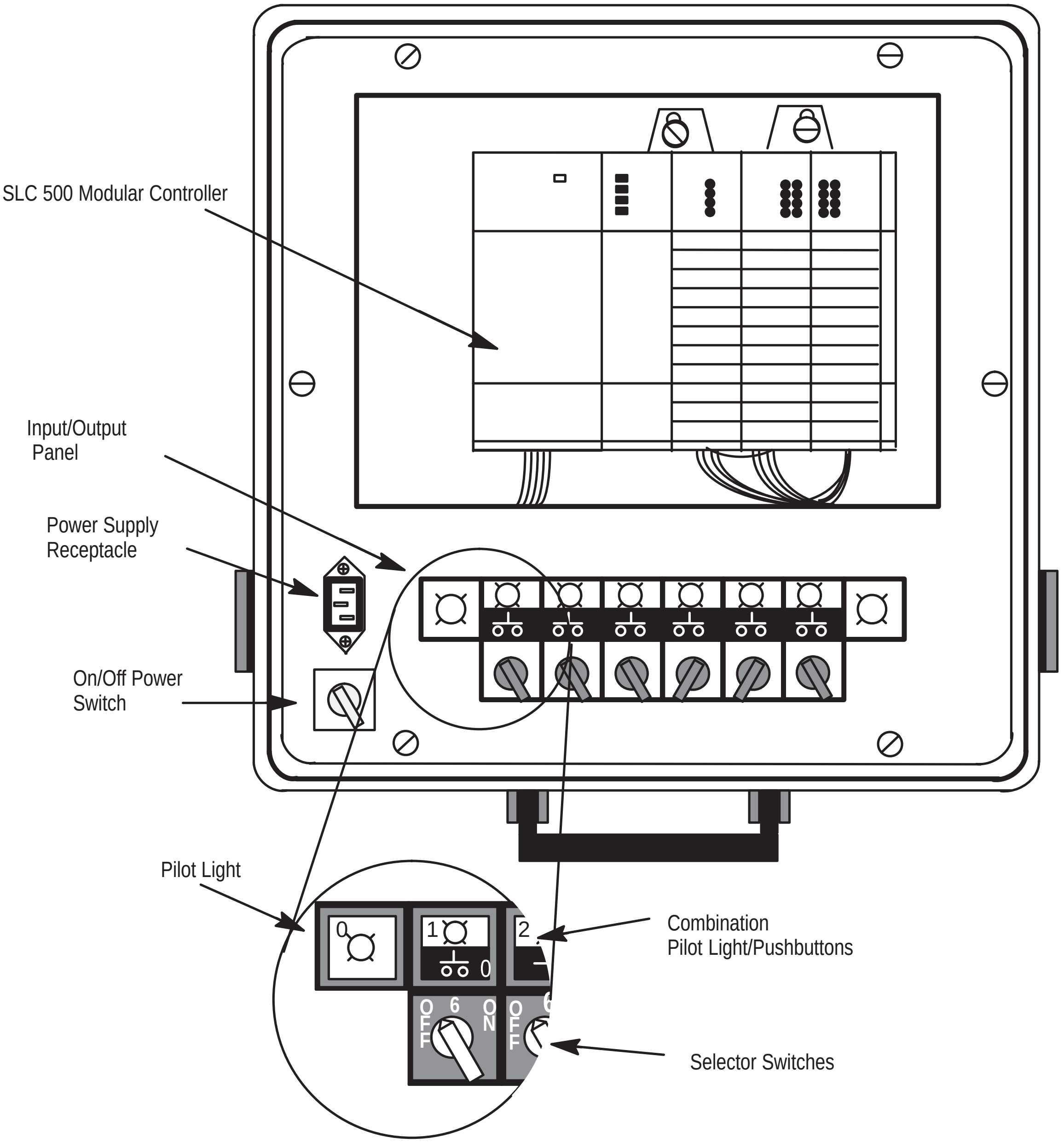
Further information on hardware is found in the Installation and Operation Manuals, Publication 1747–800 (fixed controllers) and 1747–804 (modular controllers).

Setting Up a Demo Unit

SLC 500 demo units are available with either a fixed controller or modular controller. This guide assumes you are using a modular controller demo unit for all the programming exercises. If you use a fixed controller demo unit, you will need to use different configuration information and I/O addresses in the exercises. This is explained later.

The figure below shows an SLC 500 modular controller demo unit. It is completely wired, with 12 external inputs (6 push buttons and 6 selector switches) and 8 external outputs (pilot lights).

Note the On/Off Power Switch and the Power Supply Receptacle on the demo. Make certain that the power switch is Off, then insert one end of the power cord into the power supply receptacle and the other end into an electrical socket.



Setting Up a Field–Wired Controller

The details of installing and wiring the controller and external input/output devices are beyond the scope of this guide.

If you are using a field–wired fixed or modular controller, refer to the Installation and Operation Manuals, Publication 1747–800 (fixed controllers) and 1747–804 (modular controllers), for information on installation and wiring of the controller and external input/output devices.

We recommend that your controller have two external input circuits and two external output circuits to complete the exercises in this guide.

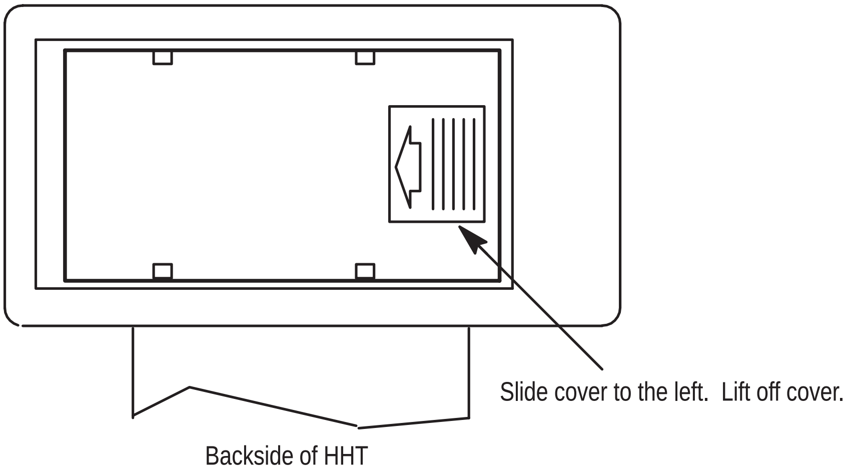
Installing the Memory Pak, Battery, and Communication Cable

The HHT (with communication cable), the memory pak, and the battery are supplied separately. Install the memory pak, battery, and communication cable as follows:

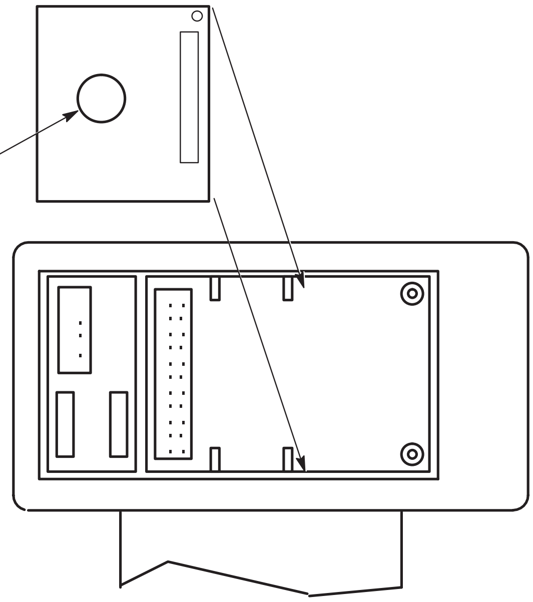
1. Install the memory pak first. The English version is catalog number 1747–PTA1E.

Important: The memory pak contains CMOS devices. Wear a grounding strap and use proper grounding procedures to guard against damage to the memory pak from electrostatic discharge.

A. To install the memory pak, remove the cover from the back of the HHT.



B. Insert the memory pak in its compartment as indicated in the following figure.



After the memory pak is in the compartment, press down on handle to secure connector in socket.

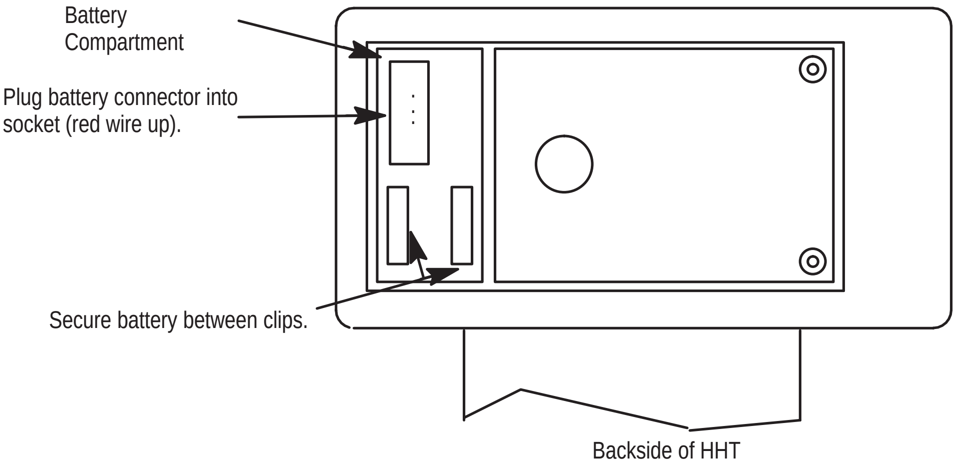
Backside of HHT

2. Install the battery, catalog number 1747–BA. The battery compartment is next to the memory pak compartment.

ATTENTION: If you do not install a battery, the letter B appears flashing on the prompt line of the HHT display to let you know that battery power is low; in addition, each time you power up, the self–test diagnostic will be interrupted, and the statement **`BATTERY TEST FAILED`** will appear.

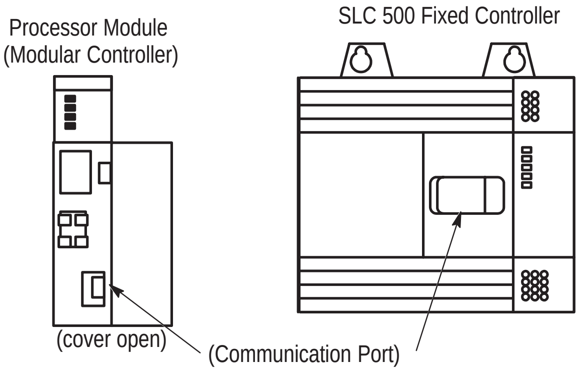
To prevent this from happening, leave the "battery low defeat jumper" inserted in the battery socket. The HHT will be functional, but your user program will be cleared form memory when you de–energize the HHT. If you do not download the user program to the processor before you de–energize the HHT, your program will be lost.

A. Remove the jumper from the battery socket, then plug the battery connector into the socket (red wire up). Secure the battery between the clips.

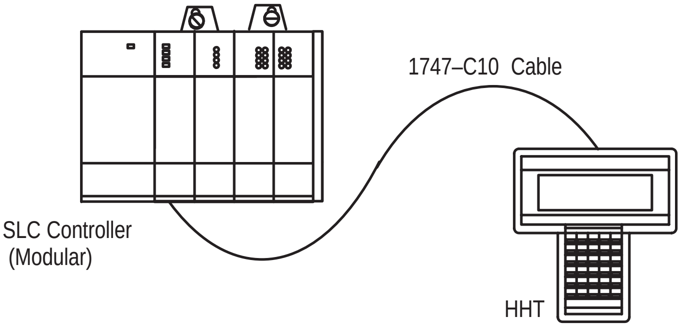


B. Replace the cover.

3. Locate the Communications Port of the controller. The figure below shows where it is located on modular and fixed controllers.



Connectors are keyed. Connect one end of the 1747–C10 communication cable to the top of the HHT. The other connector plugs into the communication port on the front of the fixed I/O controller, or into the communication port on the front of the CPU module of modular controllers.



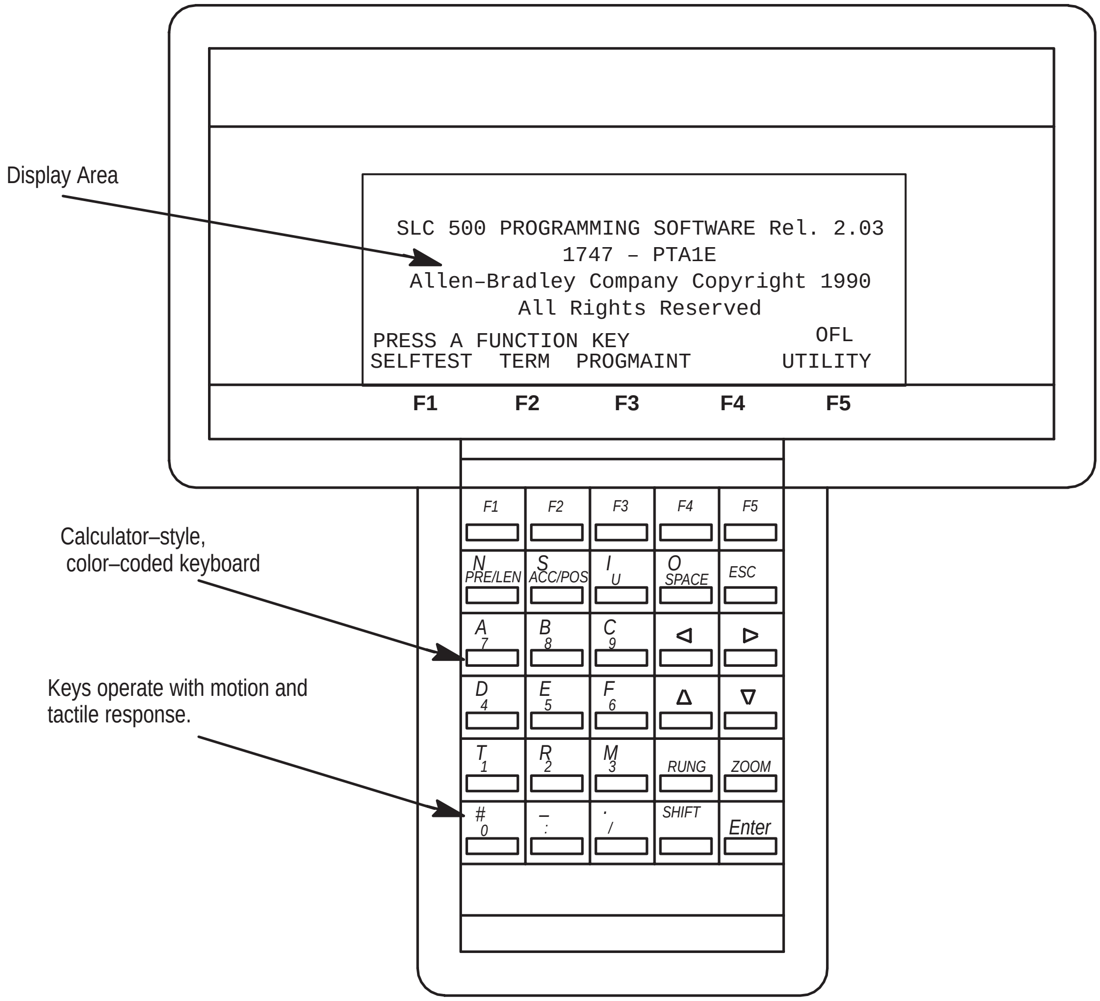
If you are using a 1747–NP1 Wall–Mount power supply or a 1747–NP2 Global Desk Top power supply, plug the communication cable connector into the socket provided.

HHT Features

Use the Hand–Held Terminal to configure the SLC 500 controller, enter/modify a user program, download/upload programs, monitor controller operation, test, and troubleshoot. The HHT has its own memory to store a program.

You can use the HHT stand alone (for remote programming development with 1747–NP1 or NP2 power supply), point–to–point (one HHT to one controller), or on a DH–485 network (communicate with up to 31 nodes over 4,000 feet). When equipped with a battery (1747–BA), the HHT will retain a user program in memory for storage and later use.

The HHT is menu–driven. The display area accommodates 8 lines x 40 characters. You can display five rungs of a user program. When monitoring the program in the RUN mode, instructions in a ladder diagram are intensified to indicate “true” status.

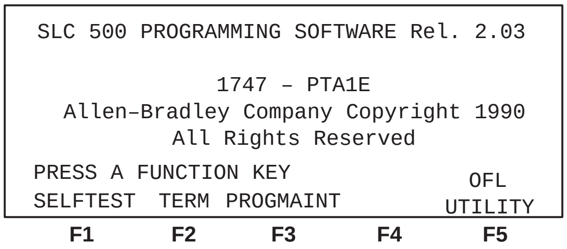


```
 SLC 500 PROGRAMMING SOFTWARE Rel. 2.03
              1747 – PTA1E
  Allen–Bradley Company Copyright 1990
          All Rights Reserved
PRESS A FUNCTION KEY                 OFL
SELFTEST  TERM  PROGMAINT       UTILITY
```

HHT Powerup

After you install the memory pak and battery, and plug in the cable, you can test operation of the HHT by powering up the controller (or plugging in the Wall–Mount or Global Desk Top power supply).

When the HHT is energized, it will go through a series of diagnostic tests. Then the following display is shown.



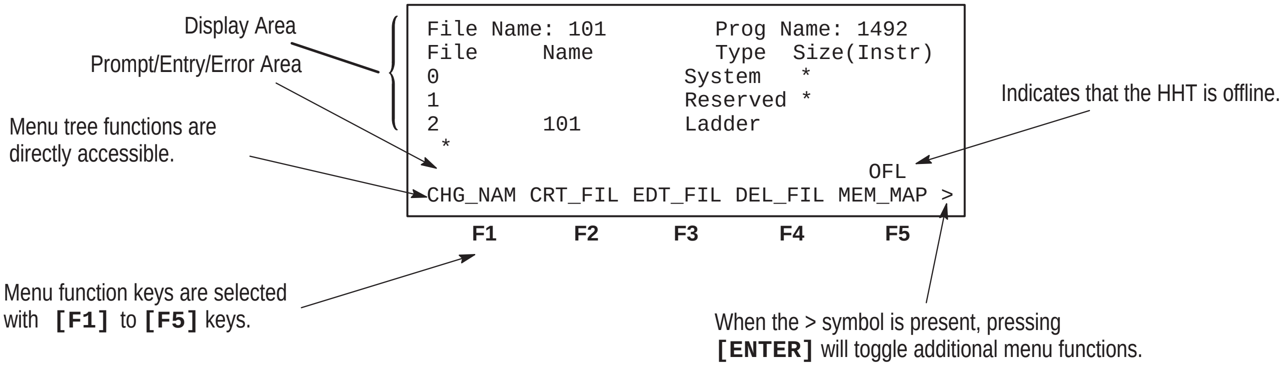
```
  SLC 500 PROGRAMMING SOFTWARE Rel. 2.03

              1747 – PTA1E
    Allen–Bradley Company Copyright 1990
            All Rights Reserved
 PRESS A FUNCTION KEY
                                     OFL
 SELFTEST  TERM  PROGMAINT          UTILITY
     F1       F2       F3       F4       F5
```

HHT Display Format

The HHT display format consists of the following:

- Display area
- Prompt/Data Entry/Error Message area
- Menu tree functions

The figure below indicates what appears in these areas. (To access this screen, press **[F3] – PROGMAINT**.)



```
File Name: 101        Prog Name: 1492
File     Name         Type  Size(Instr)
0                   System   *
1                   Reserved *
2        101        Ladder
 *
                                  OFL
CHG_NAM CRT_FIL EDT_FIL DEL_FIL MEM_MAP >
    F1       F2       F3       F4       F5
```

Display Area

Prompt/Entry/Error Area

Menu tree functions are directly accessible.

Indicates that the HHT is offline.

Menu function keys are selected with **[F1]** to **[F5]** keys.

When the > symbol is present, pressing **[ENTER]** will toggle additional menu functions.

The Keyboard

This section is intended only as a brief preview of keyboard operation. Beginning in chapter 3, you will become familiar with the keyboard as you are guided through various programming procedures.

Menu Function Keys (F1, F2, F3, F4, F5)

The top row of purple keys, F1 through F5, are menu function keys. They select the menu functions at the bottom of the screen. Note that when the > symbol is present, the **[ENTER]** key will toggle additional menu functions (if any) at a particular menu level. The **[ESC]** key exits the display to the previous menu level.

Data Entry Keys (A 7, B 8, C 9...)

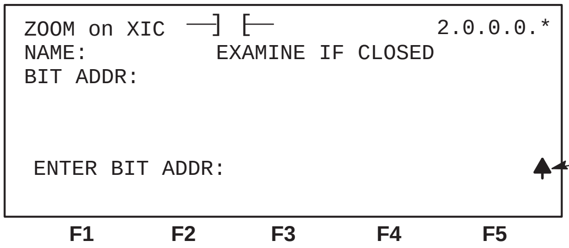
These blue keys include numbers, letters, and symbols used for addresses, password, file numbers, and other data. The data you enter always appears on the prompt/data entry/error message area of the display.

In general, you obtain the upper character of a key by pressing the **[SHIFT]** key first. You do not have to hold it and press next key. Just press and release **[SHIFT]** and then press the next key.

If you make an error while entering data, press **[ESC]** and re–enter the data, or use cursor keys and the space key. To complete a data entry, press **[ENTER]**. You can also use the **[ESC]** key to exit the data entry and return to the next higher (previous) menu level.

Auto Shift

When you enter an instruction address, the HHT automatically goes to SHIFT mode. This mode is indicated by a small arrow in the bottom right hand corner of the display.



```
ZOOM on XIC  —] [—              2.0.0.0.*
NAME:          EXAMINE IF CLOSED
BIT ADDR:



 ENTER BIT ADDR:                          ▲
```

Indicates that the HHT is in SHIFT mode (e.g., to enter the letter “I” you do not have to first select SHIFT).

Cursor Keys ∆ ,◁ ,▷ ,▽

Use these four green, arrow keys to:

- Correct data entry errors (either type over or use space key).
- Move the cursor left, right, up, and down in a ladder program. Rungs not shown in the HHT display will automatically scroll into view as you cursor down (or up) in the program.
- Scroll through controller and I/O configuration selections.
- Scroll through program file directories.
- Scroll through the elements and bits of individual data files.

ZOOM and RUNG Keys

The **[ZOOM]** key brings up a display that shows the parameters of an instruction (helpful with timers, counters, sequencers, etc.; also helpful during editing).

The **[RUNG]** key moves the cursor to a particular rung. Using the **[RUNG]** key saves time when you have a long ladder diagram. When you press **[RUNG]**, you are prompted for the rung number that you want to edit or monitor. You enter the rung number and press **[ENTER]**, then the cursor moves to the selected rung and the rung appears at the top of the screen.

Chapter 2

Control Basics

This chapter introduces you to basic concepts essential for understanding how the SLC 500 controller operates. It covers:

- SLC 500 file concepts
- How external I/O devices communicate with the processor
- Addressing external I/O
- External I/O addressing formats
- Ladder logic concepts

SLC 500 File Concepts

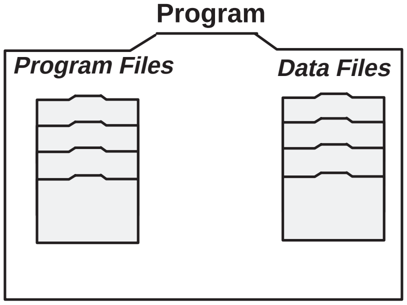
The CPU, or processor, provides control through the use of a program you create. This program contains files that break down into more manageable sections. These sections are:

- Program Files — provide storage and control of the main program and subroutines.
- Data Files — contain the status of inputs, outputs, the processor, timers, counters, and so on.

Notes on terminology: The term *program* used in Hand–Held Terminal (HHT) displays and documentation is equivalent to the term *processor file* used in APS software displays and documentation. Both terms mean the collective program files and data files created under a particular program or processor file name.

Program

The HHT and each CPU can hold one program at a time. The program is made up of program files (up to 256 per controller) and data files (up to 256 per controller).



A program is created in the offline mode using your HHT. It is then downloaded to the processor for online operation.

Program Files

Program files contain controller information, the main control program, and any subroutine programs. The first three program files are required for each program. These are:

- **File 0**
  This file stores the controller configuration and other system information.
- **File 1**
  This file is reserved for internal controller use.
- **File 2**
  This file stores the main control program.
- **Files 3 – 255**
  These files are optional and used for subroutine programs.

Most of your work with program files will be in file 2, the main program file. This file contains your ladder logic program that you create to control your application.

Data Files

Data files contain the data associated with the program files. Each program can contain up to 256 data files. These files are organized by the type of data they contain. Each piece of data in each of these files has an address associated with it that identifies it for use in the program file. For example, an input point has an address that represents its location in the input data file. Likewise, a timer in the timer data file has an address associated with it that allows you to represent it in the program file.

The first 9 data files (0 – 8) have default types. You designate the remainder of the files (9 – 255) . The default types are:

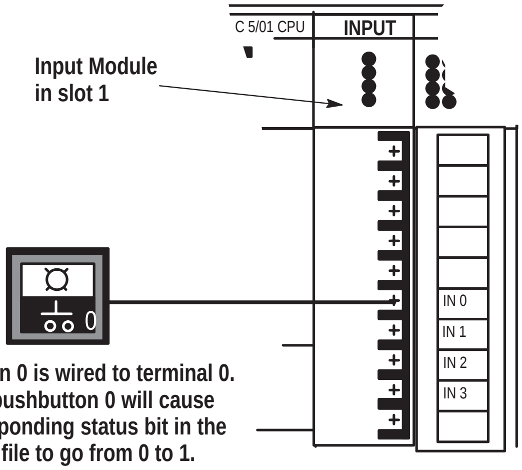
- **File 0 – Output Data**
  This file stores the state of the output terminals for the controller.
- **File 1 – Input Data**
  This file stores the status of the input terminals for the controller.
- **File 2 – Status Data**
  This file stores controller operation information.
- **Files 3 – 7**
  These files are pre–defined as Bit, Timers, Counters, Control, and Integer data storage, respectively.
- **File 8**
  This file is reserved for internal use.
- **Files 9 – 255**
  These files are user–defined as Bit, Timer, Counters, Control, and Integer data storage.

Most of your work with data files will be in files 0 and 1, the output and input files. Refer to appendix A for an example of the Timer data file.

How External I/O Devices Communicate with the Processor

The figure below applies to a modular controller demo unit having an input module in slot 1 and an output module in slot 3. See page 1–2 for a diagram of the slot location. To simplify the illustration, only pushbutton 0 and pilot light 0 of the external I/O are shown.

Each of the external input circuits is represented by a status bit in the input data file of the program. Each of the external output circuits is represented by a status bit in the output data file of the program. During controller operation, the processor applies the input data to the program, solves the program based on the instruction you enter, and energizes and de–energizes external outputs.



Input Module in slot 1

Pushbutton 0 is wired to terminal 0. Pressing pushbutton 0 will cause the corresponding status bit in the input data file to go from 0 to 1.

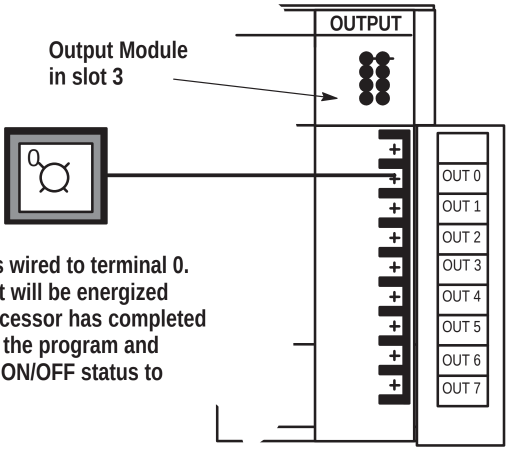
Input Data File

```
address          data
 I:1              0000
```

Status bit 0 corresponds to terminal 0 of the input module in slot 1.

Closing an external input circuit changes the corresponding status bit from 0 to 1.

Opening an external input circuit changes the corresponding status bit from 1 to 0.



Output Module in slot 3

Pilot light 0 is wired to terminal 0. The pilot light will be energized when the processor has completed evaluation of the program and transfers the ON/OFF status to the outputs.

Output Data File

```
address          data
 O:3          0000 0000
```

Status bit 0 corresponds to terminal 0 of the output module in slot 3.

When an output data file status bit has been solved as a 1, the corresponding external output circuit will be energized (ON).

When an output data file status bit has been solved as a 0, the corresponding external output circuit is de–energized (OFF).

Addressing External I/O

As pointed out in the last section, external inputs and outputs are linked to the input data file and output data file of the program. Each status bit in these files has an address. You specify the appropriate address when you enter an instruction in your ladder program.

For our purposes, input addresses have the form **`I:e/b`**
where
`I` = Input data file
`:` = Element or slot delimiter
`e` = Slot number of the input module
`/` = Bit or terminal delimiter
`b` = Terminal number used with input device

Similarly, output addresses have the form **`O:e/b`**
where
`O` = Output data file
`:` = Element or slot delimiter
`e` = Slot number of the output module
`/` = Bit or terminal delimiter
`b` = Terminal number used with output device

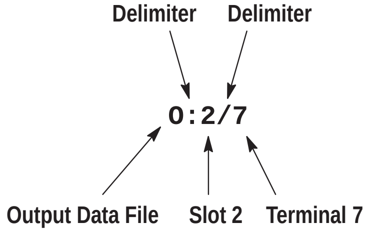
Examples:
`I:1/0` = Input, slot 1, terminal 0
`I:2/0` = Input, slot 2, terminal 0
`O:3/0` = Output, slot 3, terminal 0
`O:3/7` = Output, slot 3, terminal 7
`O:0/7` = Output, slot 0, terminal 7 (fixed controllers only because of slot 0)
`I:0/4` = Input, slot 0, terminal 4 (fixed controllers only because of slot 0)

Eventually, you will be addressing other data files, such as Status, Bit, Timer, Counter, Integer, and Control. Addressing of these files is discussed in the HHT User Manual.

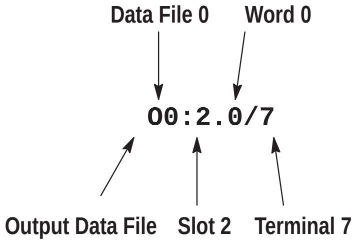
External I/O Addressing Formats

There are three ways in which an external I/O address appears in this guide:

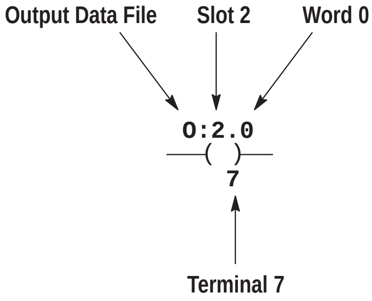
- The five keyboard entries you make to enter the address in the HHT:



- The full address, as it appears in the HHT displays:

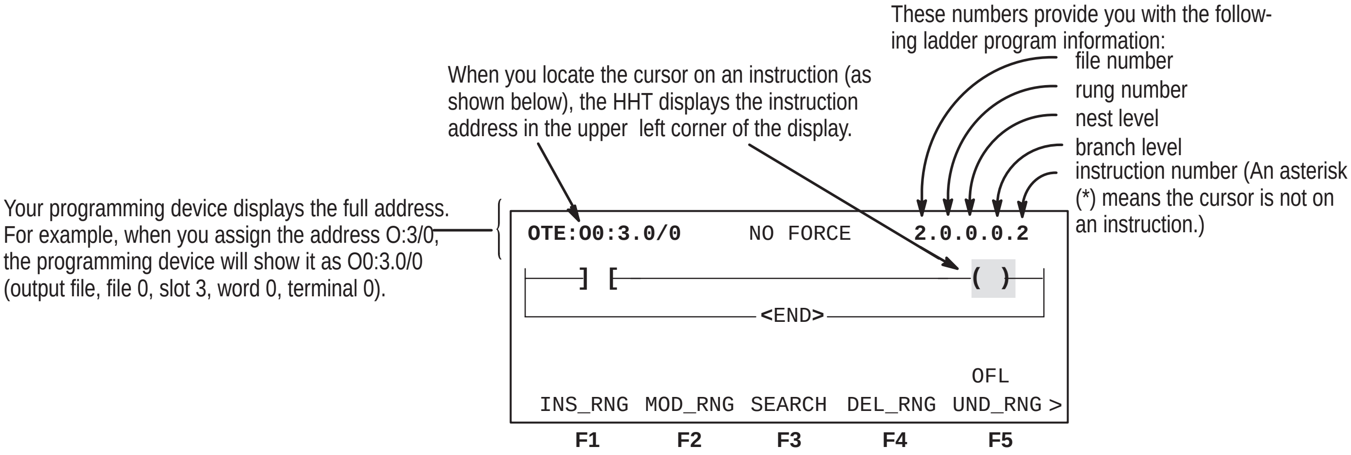


- The convention we use to show the address in the ladder diagrams:



HHT Display of Instructions/Addresses

The HHT displays I/O addresses as shown below.



These numbers provide you with the following ladder program information:

file number
rung number
nest level
branch level
instruction number (An asterisk (*) means the cursor is not on an instruction.)

When you locate the cursor on an instruction (as shown below), the HHT displays the instruction address in the upper left corner of the display.

Your programming device displays the full address. For example, when you assign the address O:3/0, the programming device will show it as O0:3.0/0 (output file, file 0, slot 3, word 0, terminal 0).

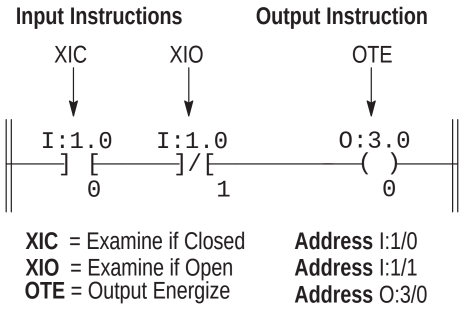
```
OTE:O0:3.0/0      NO FORCE      2.0.0.0.2
───] [──────────────────────────────( )──
────────────────<END>────────────────────

                                   OFL
 INS_RNG  MOD_RNG  SEARCH  DEL_RNG  UND_RNG >
   F1       F2       F3       F4       F5
```

Ladder Logic Concepts

As we mentioned earlier, the program files you create contain the program used for your controlling application. The programs are written in a programming language called Ladder Logic. This name is derived from its ladder–like appearance.

A ladder logic program consists of a number of rungs, on which you place instructions. Instructions each have a data address associated with them and based on the status of these instructions the rung is solved.

The figure below shows a simple 1–rung ladder program. The rung includes two input instructions and an output instruction. Note, in the example below each instruction has a name (Examine if Closed), a mnemonic (XIC), and an address (I:1/0).



A simple rung, using bit instructions.

True/False Status

The data file bits that these instructions are addressed to will be either a logic 0 (OFF) or a logic 1 (ON). This determines whether the instruction is regarded as “true” or “false”:

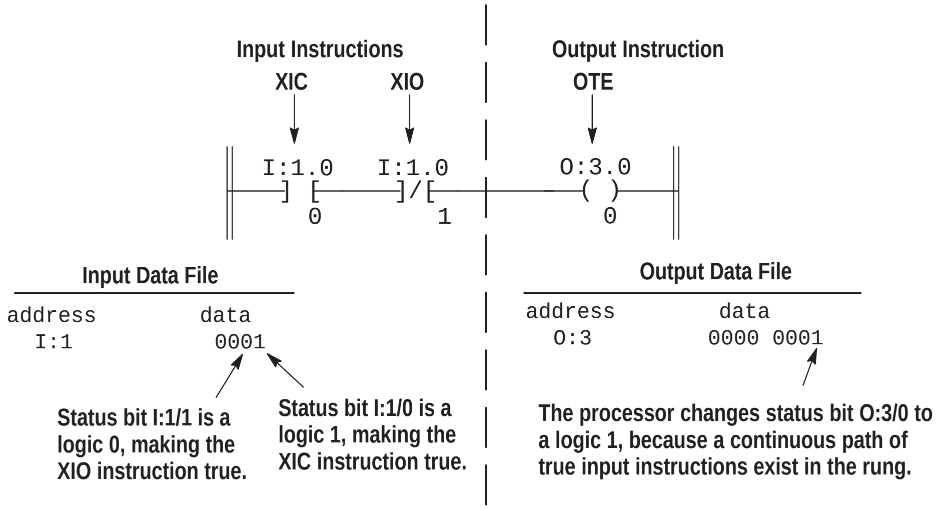
If the data file bit is	The status of the instruction is		
	XIC Examine if Closed —] [—	XIO Examine if Open —]/[—	OTE Output Energize —( )—
Logic 0	False	True	False
Logic 1	True	False	True

Logical Continuity

During controller operation, the processor evaluates each rung, changing the status of instructions according to the logical continuity of rungs. More specifically, input instructions set up the conditions under which the processor will make an output instruction true or false. These conditions are:

- When the processor finds a continuous path of true input instructions in a rung, the OTE output instruction will become (or remain) true. We then say that “rung conditions are true.”
- When the processor does *not* find a continuous path of true input instructions in a rung, the OTE output instruction will become (or remain) false. We then say that “rung conditions are false.”

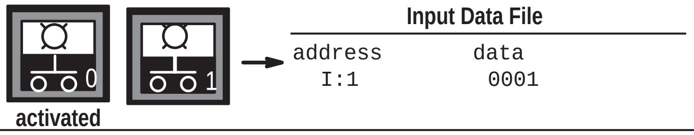
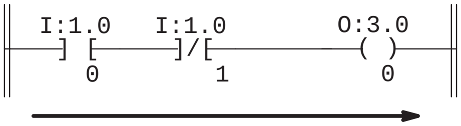
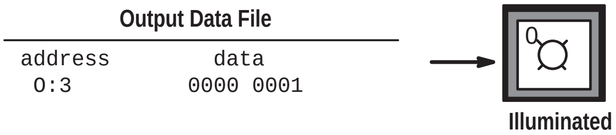
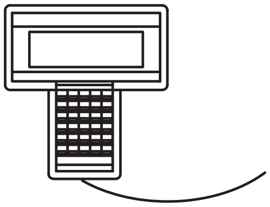
The figure below indicates the data file conditions under which the rung is true:


Input Instructions
XIC
XIO
Output Instruction
OTE
I:1.0
] [
0
I:1.0
]/[
1
O:3.0
( )
0
Input Data File
address
I:1
data
0001
Output Data File
address
O:3
data
0000 0001
Status bit I:1/1 is a logic 0, making the XIO instruction true.
Status bit I:1/0 is a logic 1, making the XIC instruction true.
The processor changes status bit O:3/0 to a logic 1, because a continuous path of true input instructions exist in the rung.

In the above example, if the input data file was 0000, then the rung would be false and the output data file would read as 0000 0000.

Processor Operating Cycle

The diagram below indicates the events that occur during the processor operating cycle. This sequence is repeated many times each second.

Event	Description	
Input Scan	 0 1 activated Input Data File address data I:1 0001	**The status of external input circuits is read. The input data file is updated with this information.**
Program Scan		**The ladder program is executed. The input data file is evaluated, the ladder rung is solved, and the output data file is updated.**
Output Scan	 Output Data File address data O:3 0000 0001 0 Illuminated	**The output data file information is transferred to the external output circuit, thus energizing or de–energizing it.**
Communications		**Communications with the HHT and other network devices takes place.**
Housekeeping	**Processor internal housekeeping takes place.**	

Creating a Program

In this chapter you create a program. The tasks you will perform:

- For modular controllers: Make a record of the processor module catalog number, the rack catalog number(s), the I/O module catalog numbers, and the slot locations of I/O modules.

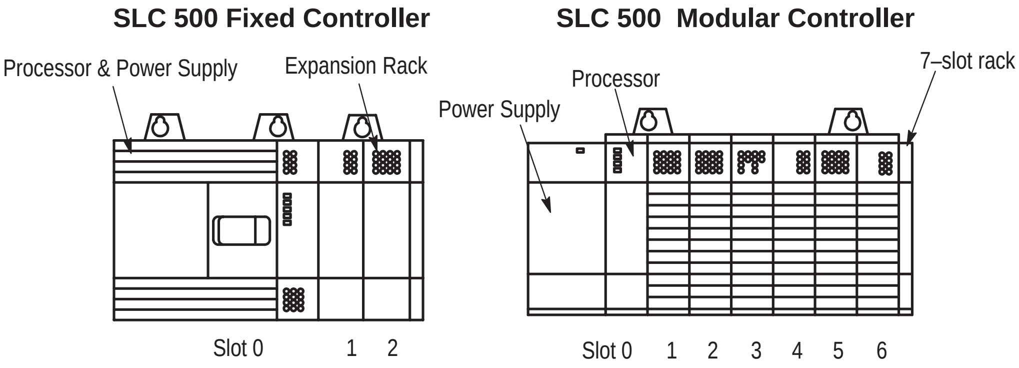
  For fixed controllers: Make a record of the controller catalog number (and I/O module catalog numbers and slot locations if you are using the 1746–A2 expansion rack).
- Use the HHT to create a program.
- Clear the HHT memory.
- Name the program “1000.”
- Enter the controller configuration.
- Enter a 1–rung ladder program.
- Save the program.

Configuration of SLC 500 Controllers

To make the best use of this guide, you should have access to an SLC 500 Demonstration Unit, which includes completely wired external inputs and outputs.   For the exercises in this guide, we assume that you are using a Demo unit using a modular controller with the components listed on page 3–4.

Controller Styles

As previously mentioned, SLC 500 controllers are available in two styles — the fixed controller and the modular controller. Examples are shown in the figure below.


SLC 500 Fixed Controller

SLC 500 Modular Controller

The fixed controller combines a power supply, processor (CPU), and a fixed number of I/O points in a single unit. You have the option of adding a 2–slot expansion rack if you want to add I/O points.

The modular controller consists of a power supply, 1–3 I/O racks, a processor module that you insert in slot 0 of the first rack, and various I/O modules that you insert in the remaining slots of the racks.

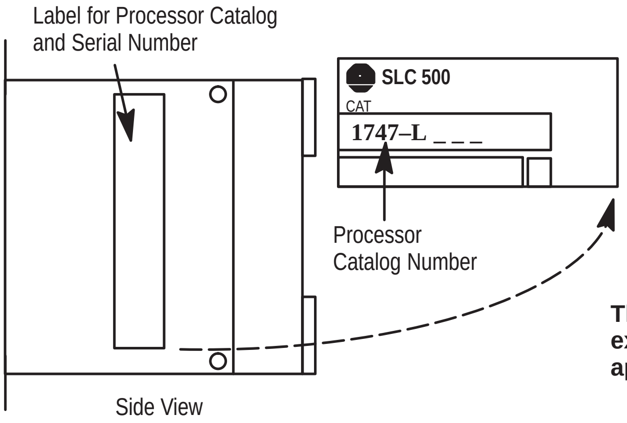
Slot Numbers: Note that slot numbers are indicated in the figure above. In fixed controllers, slot 0 applies to the processor and fixed I/O points; slots 1 and 2 apply to I/O modules located in the expansion rack. In modular controllers, slot 0 is always reserved for your processor module; the remaining slots apply to the various I/O modules you have inserted.

Catalog Numbers

When you configure your controller, you must specify the processor catalog number, rack catalog numbers, and I/O module catalog numbers as required. The location of the catalog number on the various components is shown in the following figures.

Make a record of controller components: We recommend that you make a list of the processor, rack, and I/O catalog numbers, and also the rack numbers assigned to the racks and the slot locations of all I/O modules. You can then refer to this list as you configure your controller.

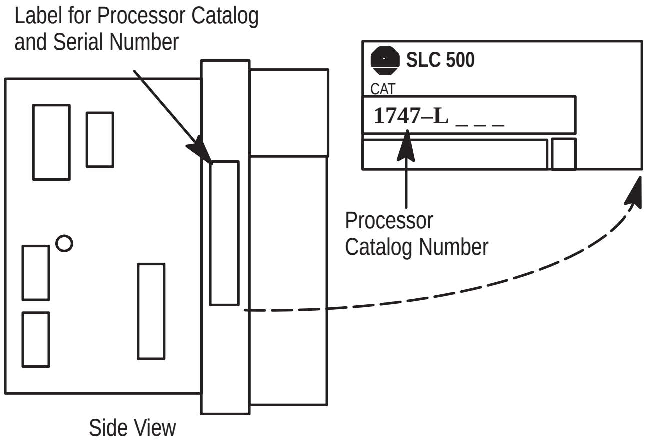
Catalog Number Location – SLC 500 Fixed Controllers



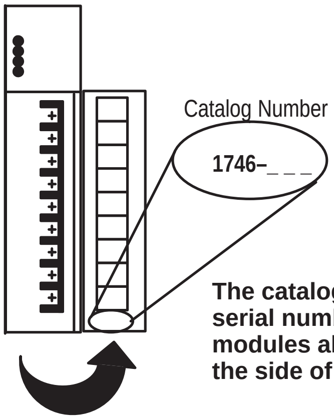
The catalog number for the expansion rack is 1746–A2. It appears on side of the rack.

Catalog Number Location – SLC 500 Modular Controllers

Processor (CPU) Modules

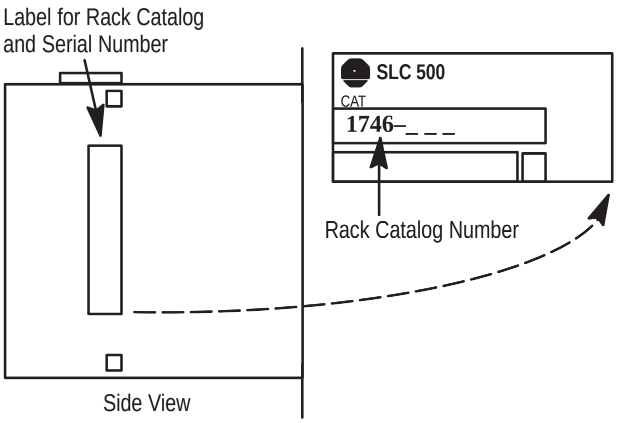


I/O Modules



The catalog number and serial number for I/O modules also appears on the side of the module.

Racks



Arbitrary Controller Used in this Guide

In the following procedures, we have assumed that the controller you are configuring in your program is a modular demo unit including the following components:

- Rack 1746–A4, 4–slot rack
- Processor 1747–L511 in slot 0
- Input module 1746–IA4 in slot 1
- Input module 1746–IA8 in slot 2
- Output module 1746–OA8 in slot 3

The ladder program shown on page 3–11 contains I/O addresses that are consistent with the configuration indicated above. If you are using some other controller configuration, keep in mind that these addresses may not be valid for your controller.

Creating a Program

A program is always created offline. In creating the program, you will:

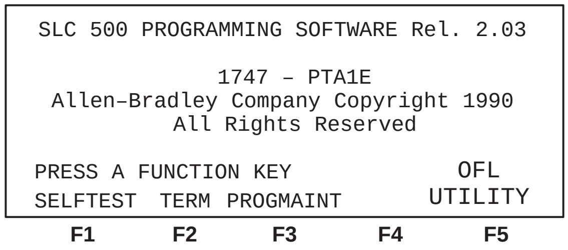
- Clear the memory of the HHT.
- Name the program and configure the controller.
- Enter a ladder program.
- Save the program in the HHT.

The abbreviated function keys that you will be encountering, both in this Guide and on the HHT display, are explained in appendix C.

Clearing the Memory of the HHT

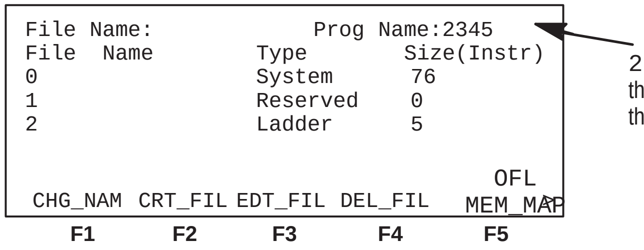
To create a new program, you must clear the HHT memory (DEFAULT program).

1. Energize your HHT. After the HHT goes through self–diagnostic tests, the following display appears:



```
 SLC 500 PROGRAMMING SOFTWARE Rel. 2.03

              1747 – PTA1E
  Allen–Bradley Company Copyright 1990
          All Rights Reserved

PRESS A FUNCTION KEY                OFL
SELFTEST  TERM  PROGMAINT         UTILITY
   F1       F2       F3       F4      F5
```

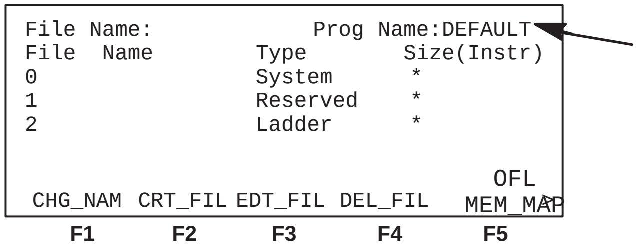
2. Press **`[F3] – PROGMAINT`**. Either the following display appears (if a program is in the HHT), or. . .



```
File Name:            Prog Name:2345
File  Name        Type       Size(Instr)
0                 System      76
1                 Reserved    0
2                 Ladder      5

                                     OFL
 CHG_NAM  CRT_FIL EDT_FIL DEL_FIL  MEM_MAP
   F1       F2       F3       F4      F5
```

`2345` (or anything other than `DEFAULT`) indicates that a program is in the HHT.

this display appears (if a program is *not* in the HHT).



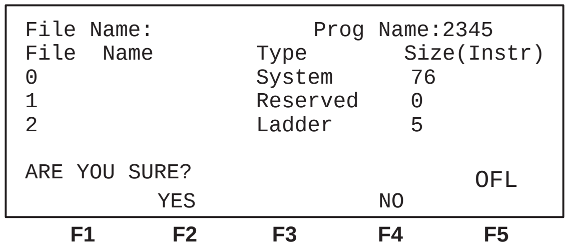
```
File Name:            Prog Name:DEFAULT
File  Name        Type       Size(Instr)
0                 System      *
1                 Reserved    *
2                 Ladder      *

                                     OFL
 CHG_NAM  CRT_FIL EDT_FIL DEL_FIL  MEM_MAP
   F1       F2       F3       F4      F5
```

`DEFAULT` indicates that a program is *not* in the HHT.

Clear the memory as follows, even if the DEFAULT program is present.

3. Press **`[ENTER]`**. More menu options appear.

4. Press **`[F4] – CLR_MEM`**. The following display appears



```
File Name:            Prog Name:2345
File  Name        Type       Size(Instr)
0                 System      76
1                 Reserved    0
2                 Ladder      5

ARE YOU SURE?                      OFL
          YES              NO
   F1       F2       F3       F4      F5
```

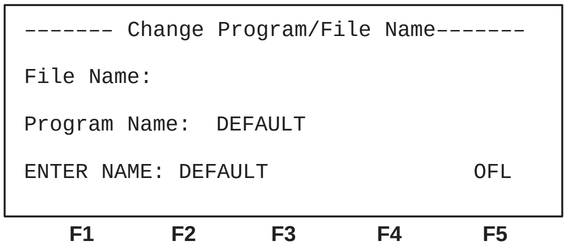
5. Press **`[F2] – YES`**. This clears the HHT memory. Now you are ready to create a program.

Naming the Program and Configuring the Controller

The following indicates how to name your program and configure your controller using the HHT.

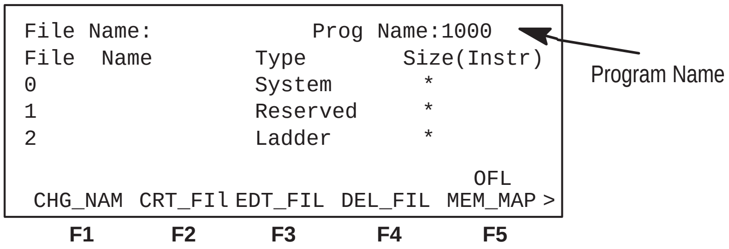
Naming Your Program

1. Press **`[ENTER]`** to view more options.
2. Press **`[F1] – CHG_NAM`**.
3. Press **`[F2] – PROGRAM`**. The following display appears:



```
------- Change Program/File Name-------

File Name:

Program Name:  DEFAULT

ENTER NAME: DEFAULT                OFL
```

4. Name your program “1000.” (Press **`[1][0][0][0][SPACE][ENTER]`**.) The program name is entered.
5. Exit this menu level by pressing **`[ESC]`**. The following display appears:



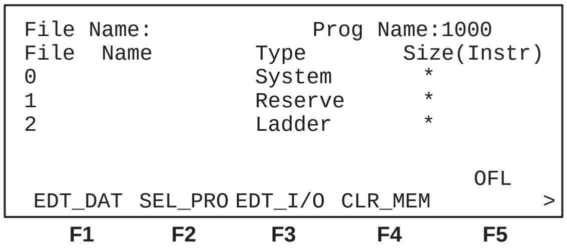
```
File Name:             Prog Name:1000
File  Name        Type       Size(Instr)
0                 System       *
1                 Reserved     *
2                 Ladder       *

                                   OFL
 CHG_NAM  CRT_FIl EDT_FIL  DEL_FIL  MEM_MAP >
```

Program Name

The program directory now shows the name of the program, which is 1000.

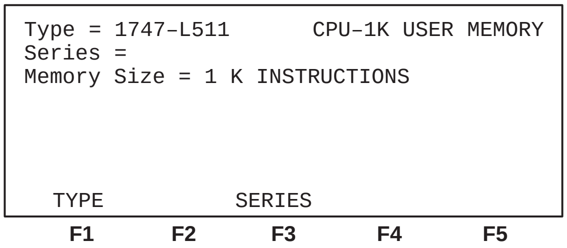
Configuring the Processor

1. Press **[ENTER]** to view the additional menu functions (as indicated by the > symbol in the lower right corner). The following display appears:



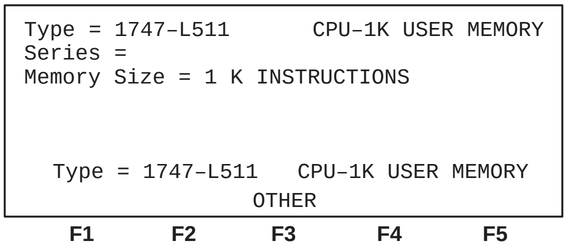
```
File Name:            Prog Name:1000
File  Name        Type       Size(Instr)
0                 System       *
1                 Reserve      *
2                 Ladder       *

                                   OFL
 EDT_DAT  SEL_PRO EDT_I/O  CLR_MEM        >
     F1      F2      F3       F4      F5
```

2. Press **[F2] – SEL_PRO**. The following display appears:



```
Type = 1747–L511      CPU–1K USER MEMORY
Series =
Memory Size = 1 K INSTRUCTIONS




  TYPE          SERIES
     F1      F2      F3       F4      F5
```

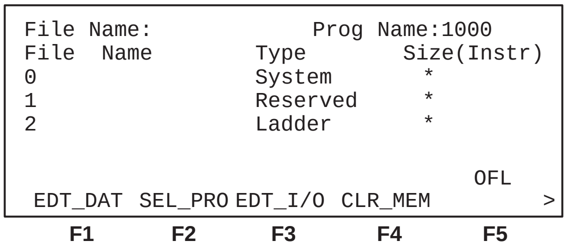
3. Press **[F1] – TYPE**. The following display appears:



```
Type = 1747–L511      CPU–1K USER MEMORY
Series =
Memory Size = 1 K INSTRUCTIONS



  Type = 1747–L511   CPU–1K USER MEMORY
                 OTHER
     F1      F2      F3       F4      F5
```

4. Use the cursor keys ([↑] or [↓]) and press **[ENTER]** to select the the right processor type. (For our example, we want the **1747–L511** type; therefore, we just press **[ENTER]**). Processor module 1747–L511 is entered into memory.

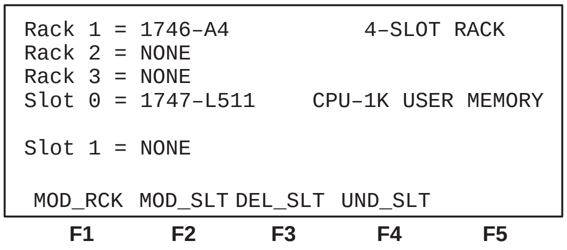
5. Press **[ESC]** to return to the following display:



```
File Name:            Prog Name:1000
File  Name        Type       Size(Instr)
0                 System       *
1                 Reserved     *
2                 Ladder       *

                                   OFL
 EDT_DAT  SEL_PRO EDT_I/O  CLR_MEM        >
     F1      F2      F3       F4      F5
```

Configuring Your I/O

1. Press **[F3] – EDT I/O**. The following display appears:

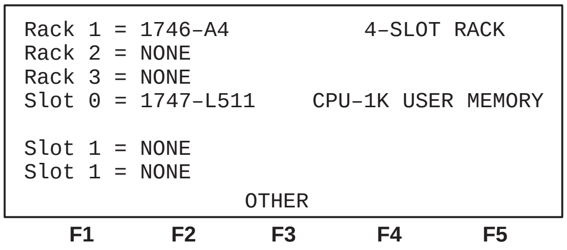


```
Rack 1 = 1746–A4          4–SLOT RACK
Rack 2 = NONE
Rack 3 = NONE
Slot 0 = 1747–L511    CPU–1K USER MEMORY

Slot 1 = NONE

 MOD_RCK  MOD_SLT DEL_SLT  UND_SLT
     F1       F2       F3       F4       F5
```

The display shows that the processor module we just entered is assigned to slot 0. It also shows the default rack selection **1746–A4**. For our example we do not have to change the rack selection. If you are using a different rack, press **[F1] – MOD_RCK**, then **[F1] – RACK 1**. Select the appropriate rack, using the up and down cursor keys, then press **[ENTER]**. If you are using more than one rack, follow the same procedure for racks 2 and 3.

The next task is to assign the I/O module slots. For our example, we use slots 1, 2 and 3.

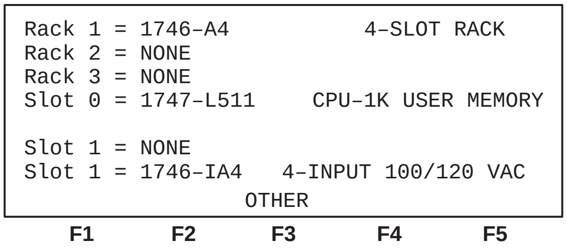
2. Press **[F2] – MOD_SLT**. The following display appears:



```
Rack 1 = 1746–A4          4–SLOT RACK
Rack 2 = NONE
Rack 3 = NONE
Slot 0 = 1747–L511    CPU–1K USER MEMORY

Slot 1 = NONE
Slot 1 = NONE
                 OTHER
     F1       F2       F3       F4       F5
```

Slot 1 = NONE appears on the prompt line.

3. Assign input module **1746–IA4** to slot 1 by scrolling or jogging with the **[↓]** key. (For our example, we press the **[↓]** key once.) The following screen appears:



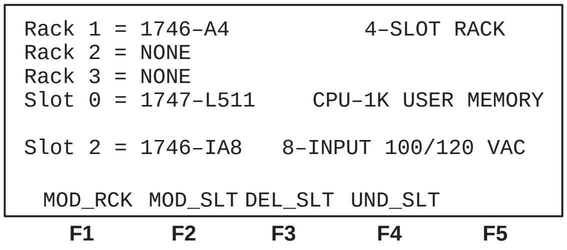
```
Rack 1 = 1746–A4          4–SLOT RACK
Rack 2 = NONE
Rack 3 = NONE
Slot 0 = 1747–L511    CPU–1K USER MEMORY

Slot 1 = NONE
Slot 1 = 1746–IA4   4–INPUT 100/120 VAC
                 OTHER
     F1       F2       F3       F4       F5
```

4. Press **[ENTER]**. **1746–IA4** is entered for slot 1.

5. Call up another slot number using the **[↓]** and **[↑]** keys. Press the **[↓]** key once.
Slot 2 = NONE appears on the prompt line.

6. Press **`[F2] – MOD_SLT`**.

7. Assign **`1746–IA8`** in slot 2: press the **[** ↓ **]** key twice, then **`[ENTER]`**. The following display appears:

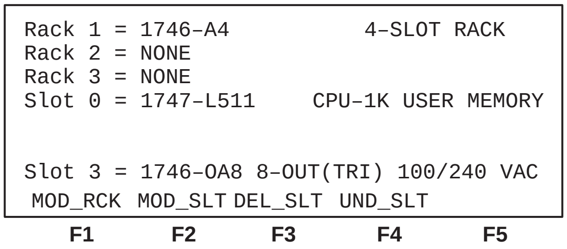


```
Rack 1 = 1746–A4          4–SLOT RACK
Rack 2 = NONE
Rack 3 = NONE
Slot 0 = 1747–L511    CPU–1K USER MEMORY

Slot 2 = 1746–IA8   8–INPUT 100/120 VAC

  MOD_RCK  MOD_SLT DEL_SLT  UND_SLT
     F1       F2      F3       F4      F5
```

8. Call up slot 3 using the cursor key. Press the **[** ↓ **]** key once.

9. Press **`[F2] – MOD_SLT`**.
`Slot 3 = NONE` appears on the prompt line.

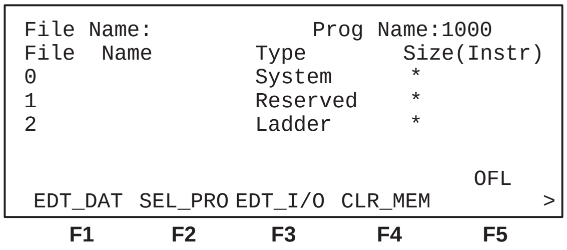
10. Assign **`1746–OA8`** in slot 3: press the **[** ↓ **]** seven times. Then press **`[ENTER]`**. The following display appears:



```
Rack 1 = 1746–A4          4–SLOT RACK
Rack 2 = NONE
Rack 3 = NONE
Slot 0 = 1747–L511    CPU–1K USER MEMORY


Slot 3 = 1746–OA8 8–OUT(TRI) 100/240 VAC
 MOD_RCK  MOD_SLT DEL_SLT  UND_SLT
     F1       F2      F3       F4      F5
```

Your controller is now fully configured.

11. Press **`[ESC]`**. This returns you to the display shown below.

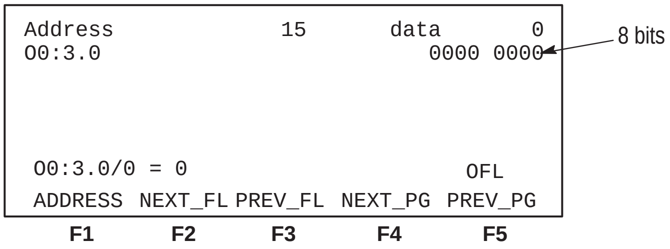


```
File Name:             Prog Name:1000
File  Name        Type        Size(Instr)
0                 System       *
1                 Reserved     *
2                 Ladder       *

                                    OFL
 EDT_DAT  SEL_PRO EDT_I/O  CLR_MEM        >
     F1       F2      F3       F4      F5
```

Monitoring Your Data File

To verify that the output and input data files for the I/O modules have been created, you can call up data files for the I/O.

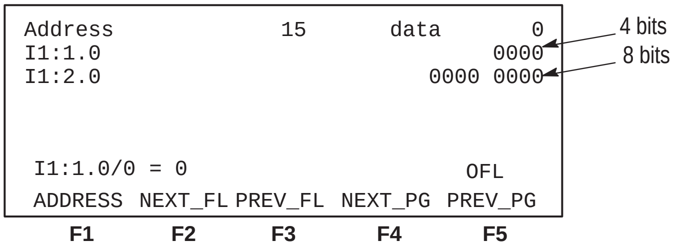
1. Press **[F1] – EDT_DAT**. The following display appears:



```
Address             15      data       0
O0:3.0                          0000 0000



 O0:3.0/0 = 0                        OFL
 ADDRESS  NEXT_FL PREV_FL  NEXT_PG  PREV_PG
```

This is file 0, the output data file. It indicates that slot 3 of the controller has 8 bits assigned (representing outputs **O:3/0** through **O:3/7**). (For more information, see the section concerning “Addressing External I/O” in chapter 2.)

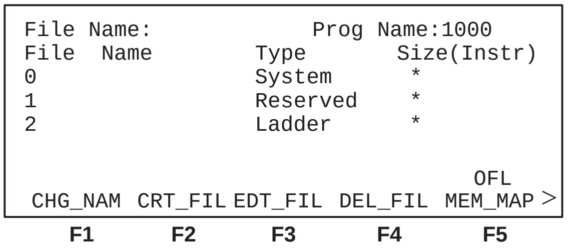
2. Press **[F2] – NEXT_FL**. The following display appears:



```
Address             15      data       0
I1:1.0                               0000
I1:2.0                          0000 0000



 I1:1.0/0 = 0                        OFL
 ADDRESS  NEXT_FL PREV_FL  NEXT_PG  PREV_PG
```

This is file 1, the input data file. It indicates that slot 1 of the controller has 4 bits assigned (representing inputs **I:1/0** through **I:1/3**), and that slot 2 has 8 bits assigned (representing inputs **I:2/0** through **I:2/7**).

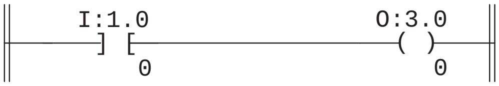
3. Now press **[ESC]**, then **[ENTER]**. The following display appears. You are now ready for the next section.



```
File Name:            Prog Name:1000
File  Name        Type       Size(Instr)
0                 System      *
1                 Reserved    *
2                 Ladder      *

                                      OFL
 CHG_NAM  CRT_FIL EDT_FIL  DEL_FIL  MEM_MAP >
```

Programming a Simple Ladder Rung

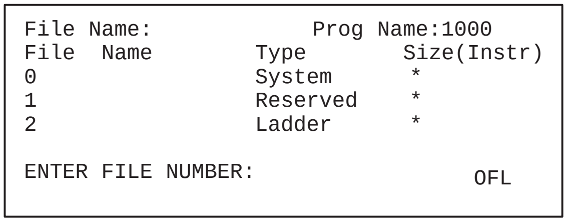
The following rung consists of an XIC input instruction and an OTE output instruction. The addresses conform to the controller configuration indicated in the “Arbitrary Controller” section of this chapter. *If you have entered a different controller configuration, make certain that the addresses are consistent with your configuration.* It is also important that you have an external input, such as a pushbutton, and an external output, such as a pilot light, at the terminal addresses used. You will be using these external devices in later chapters of this guide.



Entering a Rung

To enter the rung, do the following:

1. Begin where we ended on the previous page, at the program directory display.
2. Press **`[F3] – EDT_FIL`**. The following display appears.

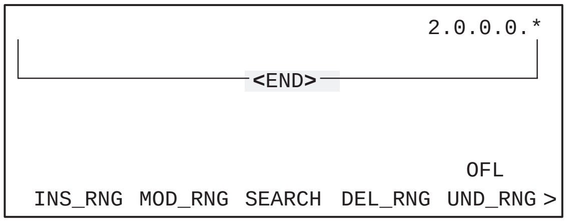


```
File Name:              Prog Name:1000
File  Name         Type        Size(Instr)
0                  System       *
1                  Reserved     *
2                  Ladder       *

ENTER FILE NUMBER:                  OFL
```

F1 F2 F3 F4 F5

We want to edit file number 2, our main program file.

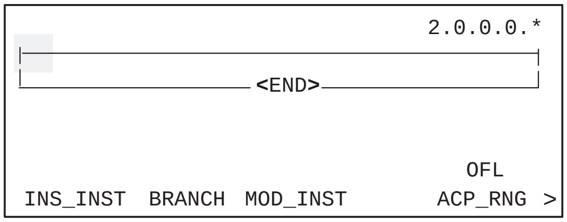
3. Press **`[2] [ENTER]`**. The display shows the END rung of a program. No other rungs exist at this time. The numbers **`2.0.0.0.*`** appear in the upper right corner of the display. This indicates that we are in File 2, and the cursor is located on rung 0, branch 0.



```
                                  2.0.0.0.*
 ├─────────────────<END>─────────────────┤




                                     OFL
 INS_RNG  MOD_RNG  SEARCH  DEL_RNG  UND_RNG >
```

F1 F2 F3 F4 F5

4. Press **`[F1] – INS_RNG`**. The following display appears:

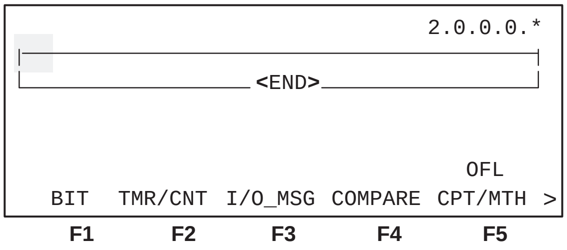


```
                                  2.0.0.0.*
 ├───────────────────────────────────────┤
 ├─────────────────<END>─────────────────┤




                                     OFL
INS_INST  BRANCH  MOD_INST        ACP_RNG  >
```

F1 F2 F3 F4 F5

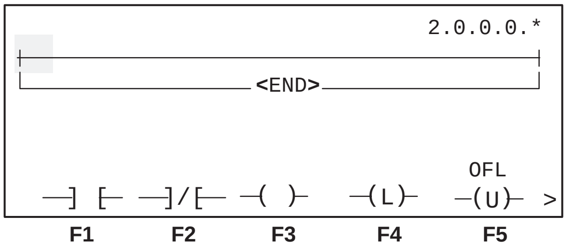
Entering an "Examine if Closed" Instruction

1. Press **[F1] – INS_INST**. The following display appears:



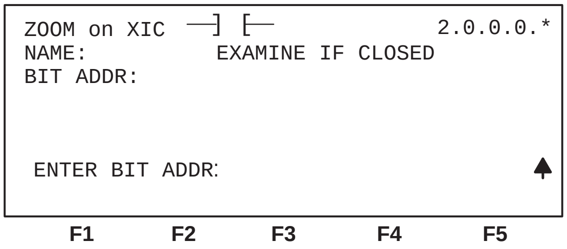
```
                                         2.0.0.0.*
 ├────────────────────────────────────────────┤
 └──────────────────<END>─────────────────────┘

                                           OFL
   BIT   TMR/CNT  I/O_MSG  COMPARE  CPT/MTH  >
     F1       F2       F3       F4       F5
```

2. Press **[F1] – BIT**. The following display appears:



```
                                         2.0.0.0.*
 ├────────────────────────────────────────────┤
 └──────────────────<END>─────────────────────┘

                                           OFL
  ─] [─  ─]/[─  ─( )─   ─(L)─   ─(U)─  >
     F1       F2       F3       F4       F5
```

3. Press **[F1] –] [–**. The following display appears:

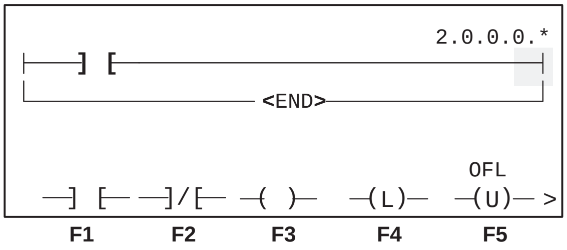


```
 ZOOM on XIC  ─] [─                 2.0.0.0.*
 NAME:          EXAMINE IF CLOSED
 BIT ADDR:



  ENTER BIT ADDR:                            ⇧
     F1       F2       F3       F4       F5
```

Note that the HHT "shifts" for you.

4. At the **ENTER BIT ADDR:** prompt, type the address **I:1/0**, then **[ENTER]**. (If you entered the wrong instruction by mistake, just press **[ESC]** and try again.)

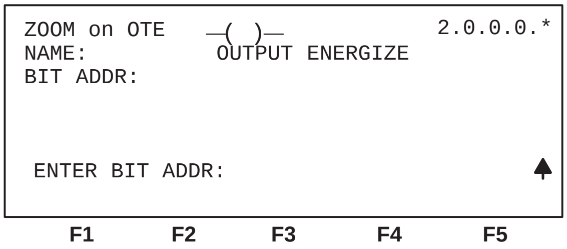
5. Press **[F5] – ACCEPT**. This accepts the address. The following display appears:



```
                                          2.0.0.0.*
 ├───] [──────────────────────────────────────┤
 └──────────────────<END>─────────────────────┘

                                           OFL
  ─] [─  ─]/[─  ─( )─   ─(L)─   ─(U)─  >
     F1       F2       F3       F4       F5
```

Entering an “Output Energize” Instruction

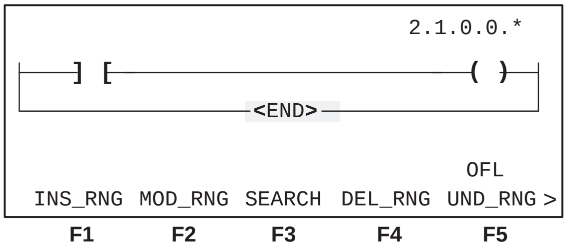
1. Press **[F3]**, for the output energize instruction. The following display appears:



```
ZOOM on OTE    —( )—                2.0.0.0.*
NAME:          OUTPUT ENERGIZE
BIT ADDR:



 ENTER BIT ADDR:                            ▲
     F1       F2       F3       F4       F5
```

2. Type bit address **O:3/0**, then **[ENTER]**. (If you entered the wrong instruction by mistake, just press **[ESC]** and try again.)

3. Press **[F5] – ACCEPT**, then press **[ESC]** twice. Then press **[F5] – ACP_ RNG**. The following display appears:



```
                                       2.1.0.0.*
├────] [──────────────────────────────( )───┤
├───────────────────<END>───────────────────┤


                                            OFL
 INS_RNG  MOD_RNG  SEARCH  DEL_RNG  UND_RNG >
     F1       F2       F3       F4       F5
```

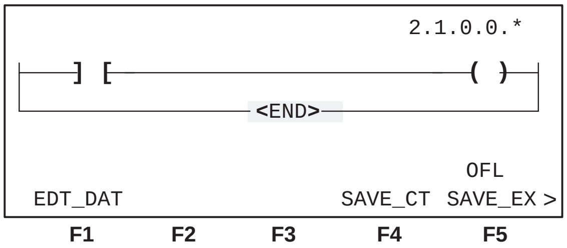
At this point the rung is entered and accepted.

4. Press **[ENTER]** to display more menu options.

Saving Your Program

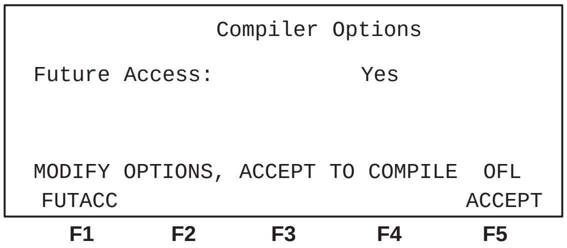
Save your program at the end of a program edit. First your program is compiled, transforming it into a more efficient package. Then the program is saved from the work area into another part of memory. In addition, the contents of program files and data files are updated. Also, a summary of data words used, instructions used, and available memory is updated.

1. Start with the screen below, where we left off in the last section.



```
                                2.1.0.0.*
 ────] [──────────────────────────( )──
                  <END>

                                     OFL
 EDT_DAT                 SAVE_CT  SAVE_EX >
     F1      F2      F3      F4      F5
```

2. Press **`[F5] – SAVE_EX`**. The following display appears:

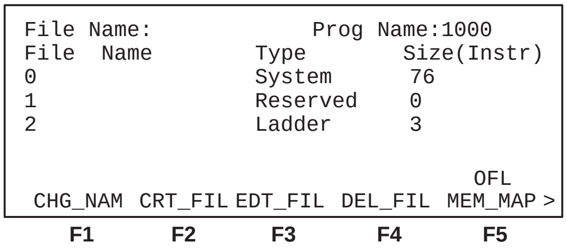


```
              Compiler Options

 Future Access:           Yes



 MODIFY OPTIONS, ACCEPT TO COMPILE  OFL
  FUTACC                          ACCEPT
     F1      F2      F3      F4      F5
```

FUTACC (Future Access): This option lets you protect proprietary program data and algorithms. The protection becomes operational only after the program is downloaded to the controller. The “Future Access: No” selection disallows online access to the processor unless a matching copy of the online program is resident in the HHT.

Important: Do not select “Future Access: No” for this exercise.

3. Press **`[F5] – ACCEPT`**. This compiles and saves the program. After short time, the following display appears:



```
File Name:              Prog Name:1000
File  Name          Type        Size(Instr)
0                   System       76
1                   Reserved     0
2                   Ladder       3

                                      OFL
 CHG_NAM  CRT_FIL EDT_FIL  DEL_FIL  MEM_MAP >
     F1      F2      F3      F4      F5
```

4. Return to the main display by pressing **`[ESC]`**.

Chapter 4

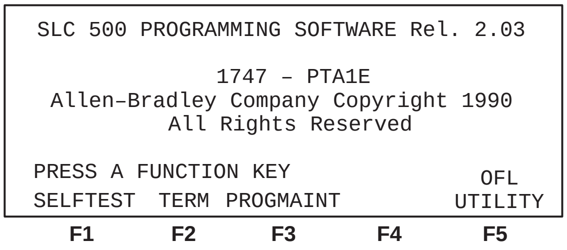
Online Operations

In this chapter, you will complete the following tasks:

- Download program 1000, created in chapter 3.
- Test the program.
- Monitor the input and output data files.

Downloading Your Program

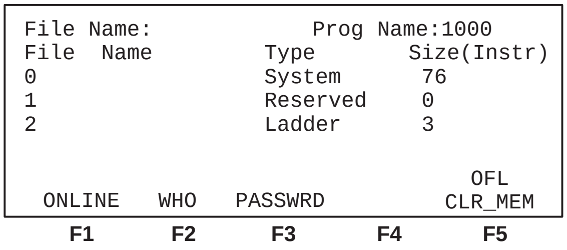
This chapter shows you how to download a program from the HHT to the processor, then monitor the program. It assumes that you have performed the tasks in chapter 3, and that the HHT shows the following display.



```
  SLC 500 PROGRAMMING SOFTWARE Rel. 2.03

               1747 – PTA1E
   Allen–Bradley Company Copyright 1990
            All Rights Reserved

 PRESS A FUNCTION KEY                   OFL
 SELFTEST  TERM  PROGMAINT          UTILITY
     F1       F2       F3       F4       F5
```

Going Online

1. Press **[F5] – UTILITY**. The following display appears:

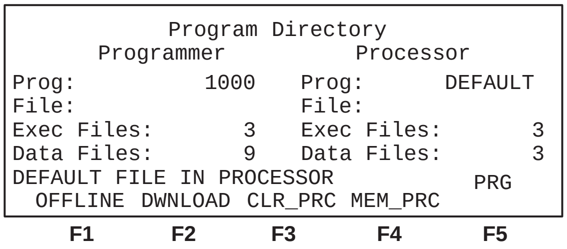


```
File Name:            Prog Name:1000
File  Name         Type       Size(Instr)
0                  System      76
1                  Reserved    0
2                  Ladder      3


                                     OFL
 ONLINE   WHO   PASSWRD            CLR_MEM
     F1       F2       F3       F4       F5
```

2. Press **[F2] – WHO**.

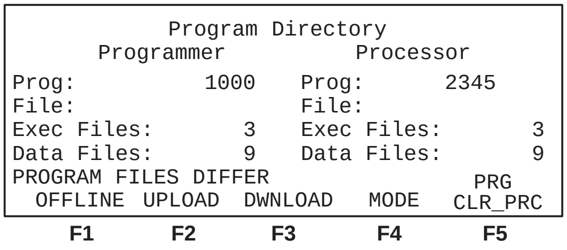
3. Press **[F3] – ATTACH**.

   Either the following display appears (if a program is not in processor memory), or. . .



```
            Program Directory
      Programmer          Processor
Prog:          1000   Prog:      DEFAULT
File:                 File:
Exec Files:       3   Exec Files:       3
Data Files:       9   Data Files:       3
DEFAULT FILE IN PROCESSOR            PRG
  OFFLINE DWNLOAD CLR_PRC MEM_PRC
     F1       F2       F3       F4       F5
```

   this display appears (if a program is in processor memory).

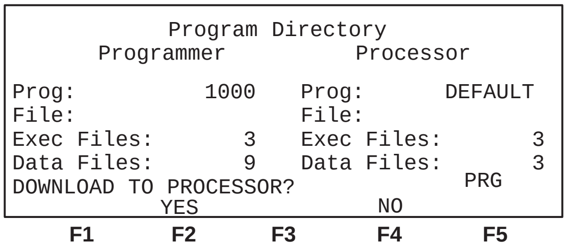


```
            Program Directory
      Programmer          Processor
Prog:          1000   Prog:      2345
File:                 File:
Exec Files:       3   Exec Files:       3
Data Files:       9   Data Files:       9
PROGRAM FILES DIFFER                 PRG
  OFFLINE  UPLOAD  DWNLOAD   MODE   CLR_PRC
     F1       F2       F3       F4       F5
```

Important: If the above display appears and the processor is not in program mode, then you must change the processor mode. To do this, press **[F4] – MODE**, then **[F5] – PROGRAM**, then **[F2] – YES**, and then **[ESC]**.

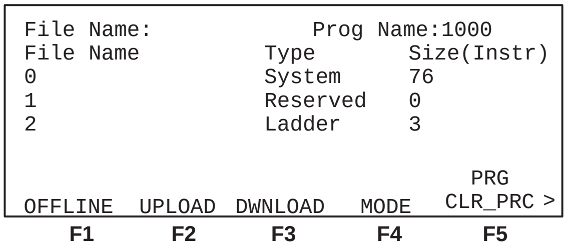
Downloading Your Program

1. Press **DWNLOAD**. The following display appears:



```
               Program Directory
        Programmer          Processor

Prog:          1000    Prog:      DEFAULT
File:                  File:
Exec Files:       3    Exec Files:       3
Data Files:       9    Data Files:       3
DOWNLOAD TO PROCESSOR?                PRG
            YES                NO
     F1      F2      F3      F4      F5
```

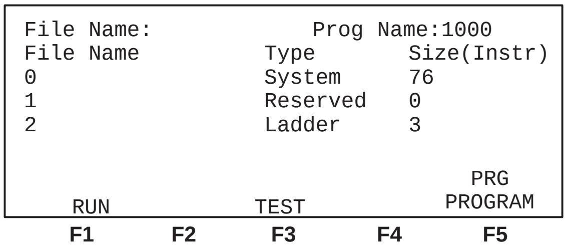
2. Press **[F2] – YES**. This verifies that you want to download the file to the processor. For a brief moment, the following message should appear **DOWNLOADING FILE.** The following display appears:



```
File Name:           Prog Name:1000
File Name          Type       Size(Instr)
0                  System     76
1                  Reserved   0
2                  Ladder     3

                                      PRG
OFFLINE  UPLOAD  DWNLOAD   MODE   CLR_PRC >
     F1      F2      F3      F4      F5
```

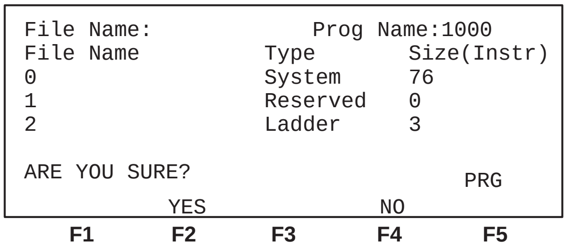
Changing Program Mode to Run Mode

1. Press **[F4] – MODE**. The following display appears:



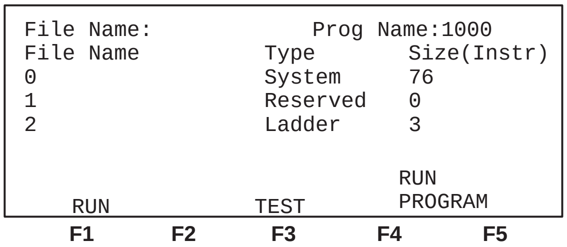
```
File Name:           Prog Name:1000
File Name          Type       Size(Instr)
0                  System     76
1                  Reserved   0
2                  Ladder     3

                                      PRG
    RUN            TEST           PROGRAM
     F1      F2      F3      F4      F5
```

2. Press **[F1] – RUN**. The following display appears:



```
File Name:           Prog Name:1000
File Name          Type       Size(Instr)
0                  System     76
1                  Reserved   0
2                  Ladder     3

ARE YOU SURE?                         PRG
            YES                NO
     F1      F2      F3      F4      F5
```

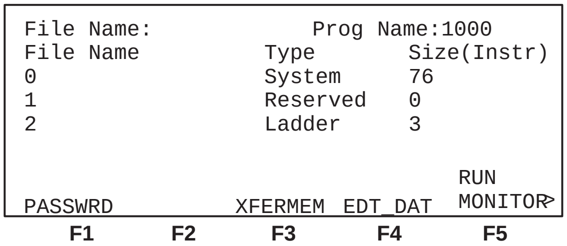
3. Press **[F2] – YES**. The following display appears again:



```
File Name:            Prog Name:1000
File Name          Type       Size(Instr)
0                  System     76
1                  Reserved   0
2                  Ladder     3

                               RUN
    RUN             TEST       PROGRAM
     F1       F2       F3       F4       F5
```

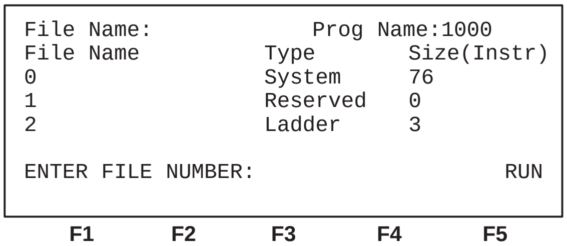
Monitoring the Program in Run Mode

1. Press **[ESC]**, then **[ENTER]**. The following display appears:



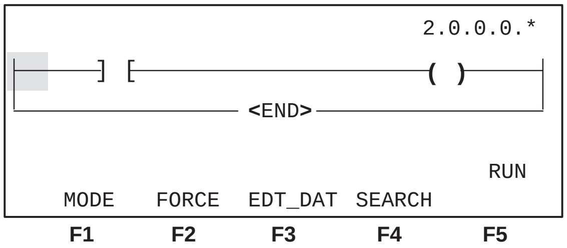
```
File Name:            Prog Name:1000
File Name          Type       Size(Instr)
0                  System     76
1                  Reserved   0
2                  Ladder     3

                                    RUN
PASSWRD          XFERMEM  EDT_DAT   MONITOR>
     F1       F2       F3       F4       F5
```

2. Press **[F5] – MONITOR**. The following display appears:



```
File Name:            Prog Name:1000
File Name          Type       Size(Instr)
0                  System     76
1                  Reserved   0
2                  Ladder     3

ENTER FILE NUMBER:                    RUN

     F1       F2       F3       F4       F5
```

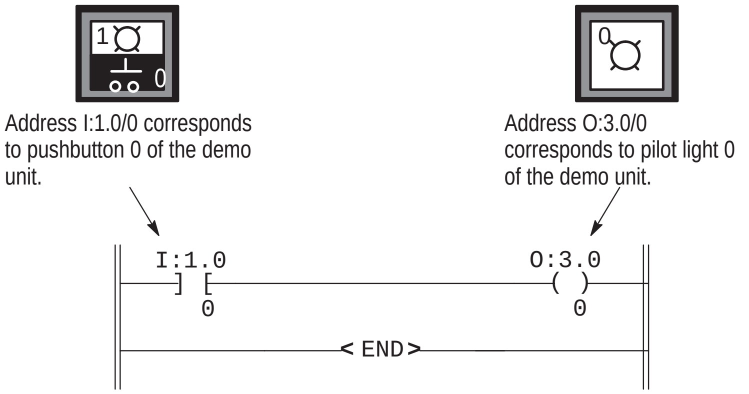
3 . Press **2**, then **[ENTER]**. The following display appears:



```
                                2.0.0.0.*
|------] [----------------------( )------|
|------------------ <END> ---------------|

                                      RUN
    MODE   FORCE   EDT_DAT  SEARCH
     F1       F2       F3       F4       F5
```

Testing Your Downloaded Program

The following diagram shows the rung you entered if you are using the modular controller demo unit discussed in chapter 3. If you are using some other controller configuration, make certain that your external input device and output device are wired to the controller input and output that you addressed in your ladder program.



Address I:1.0/0 corresponds to pushbutton 0 of the demo unit.

Address O:3.0/0 corresponds to pilot light 0 of the demo unit.

To test the program, press pushbutton 0. Pilot light 0 should go on. The display should show both the XIC and OTE instructions highlighted to indicate that they are true.

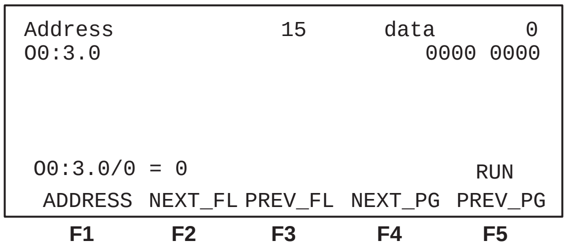
Processor operation: When you pressed pushbutton 0, the input instruction went from false to true. This resulted in a path of true input instructions in the rung, causing the output instruction to go from false to true.

Now, release the pushbutton. Pilot light 0 should go off. Neither instruction in the rung should be highlighted. When you released pushbutton 0, the input instruction went from true to false; this broke the path of true input instructions, causing the output instruction to go from true to false.

Monitoring Data Files

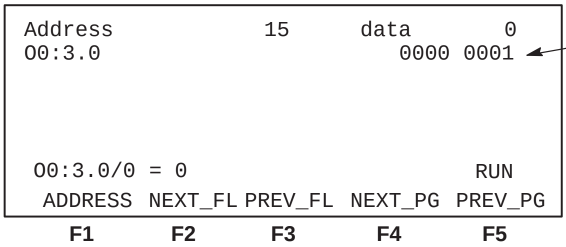
In this procedure, you will monitor the input data file and the output data file. These files include a status bit for each of the configured I/O terminals of the controller. You will monitor data file changes as you operate pushbutton 0. To end the exercise, you will go offline.

1. Beginning with the display shown on the previous page, press **[F3] – EDT_DAT**. The following display appears:



```
Address              15      data       0
O0:3.0                          0000 0000



 O0:3.0/0 = 0                        RUN
  ADDRESS  NEXT_FL PREV_FL  NEXT_PG  PREV_PG
     F1       F2      F3      F4       F5
```

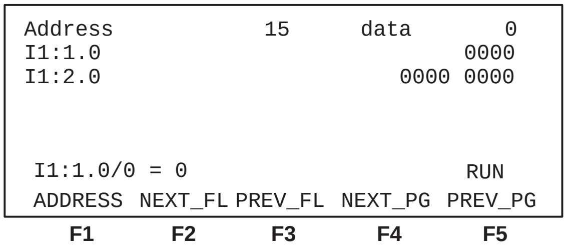
2. Monitor output data changes resulting from input device operation. Press pushbutton 0. Note that the status bit corresponding with output **`O:3/0`** goes from 0 to 1, as the instruction goes from false to true.



```
Address            15      data       0
O0:3.0                       0000 0001


 O0:3.0/0 = 0                      RUN
  ADDRESS  NEXT_FL PREV_FL  NEXT_PG  PREV_PG
    F1        F2      F3       F4       F5
```

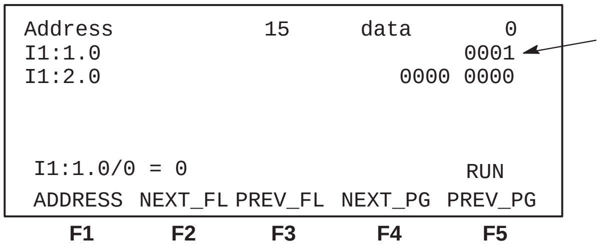
Bit changes from 0 to 1.

3. Press **`[F2] – NEXT_FL`**. The following display appears:



```
Address            15      data       0
I1:1.0                            0000
I1:2.0                       0000 0000


 I1:1.0/0 = 0                     RUN
 ADDRESS  NEXT_FL PREV_FL  NEXT_PG  PREV_PG
    F1       F2      F3       F4       F5
```

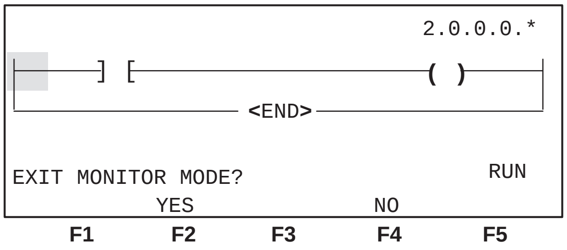
4. Monitor input data changes resulting from input device operation. Press pushbutton 0. Note that the status bit corresponding to input **`I:1/0`** goes from 0 to 1, as the instruction goes from false to true.



```
Address            15      data       0
I1:1.0                            0001
I1:2.0                       0000 0000


 I1:1.0/0 = 0                     RUN
 ADDRESS  NEXT_FL PREV_FL  NEXT_PG  PREV_PG
    F1       F2      F3       F4       F5
```

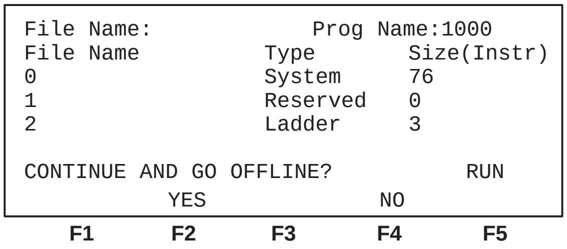
Bit changes from 0 to 1.

5. Return to the main display. Press **`[ESC]`** twice. The following display appears:



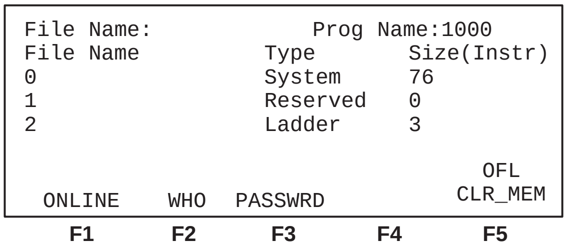
```
                                  2.0.0.0.*

|------] [------------------------( )------|
|_________________ <END> __________________|


EXIT MONITOR MODE?                   RUN
           YES              NO
    F1       F2      F3       F4       F5
```

6. Press **`[F2] – YES`**, then press **`[ESC]`** once. The following display appears:



```
File Name:            Prog Name:1000
File Name          Type       Size(Instr)
0                  System     76
1                  Reserved   0
2                  Ladder     3

CONTINUE AND GO OFFLINE?          RUN
           YES              NO
    F1       F2      F3       F4       F5
```

7. Press **[F2] – YES**. The following screen appears:



```
File Name:           Prog Name:1000
File Name          Type       Size(Instr)
0                  System     76
1                  Reserved   0
2                  Ladder     3

                                   OFL
 ONLINE    WHO  PASSWRD          CLR_MEM
```

8. Press **[ESC]**. This brings up the main display.

Appendix A

Additional Ladder Program Exercises

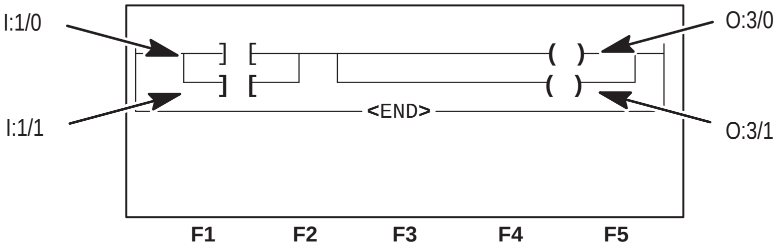
This appendix lets you apply what you have learned in the previous chapters. It covers:

- Entering a program with input and output branches
- Entering a program with a timer instruction

Entering Input and Output Branches

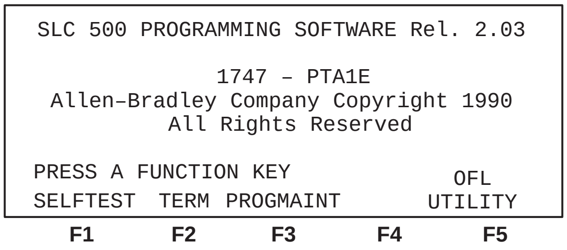
The important feature of this program is the output and input branch. The input branch is based on what is called OR or parallel logic. This means that if either input #0 OR input #1 is true, then output #0 and #1 turn on. If the rung is true, all outputs will be energized. If the rung is false, all outputs will be de–energized.

We will be editing the program created in chapter 3 to look like the one shown in the display below. It will consist of one input branch and one output branch.



Creating the Program

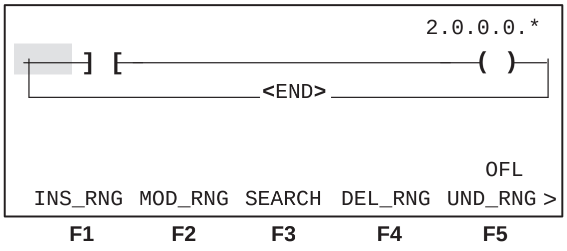
1. Start from the main display, which you ended with in chapter 4, Online Operations.



```
 SLC 500 PROGRAMMING SOFTWARE Rel. 2.03

               1747 – PTA1E
  Allen–Bradley Company Copyright 1990
           All Rights Reserved

 PRESS A FUNCTION KEY                 OFL
 SELFTEST  TERM  PROGMAINT          UTILITY
    F1       F2      F3       F4       F5
```

2. Press the following function keys (in order): **`[F3] – PROGMAINT`**, then **`[F3] – EDT_FIL`**.

3. Enter the file number, which is **`2`**. Then press **`[ENTER]`**. The program entered in chapter 3 is displayed.

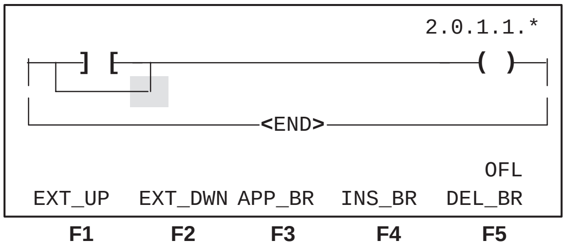


```
                                   2.0.0.0.*
 ────] [──────────────────────────( )──
 └──────────────<END>─────────────────┘


                                      OFL
 INS_RNG MOD_RNG SEARCH DEL_RNG UND_RNG >
    F1      F2      F3     F4      F5
```

4. Press **`[F2] – MOD_RNG`**, then **`[F2] – BRANCH`**.

Adding an Input Branch

5. Press **`[F4] – INS_BR`**. Then press the **[→]** cursor key once, then press **`[ENTER]`**. Your new display should look like this:

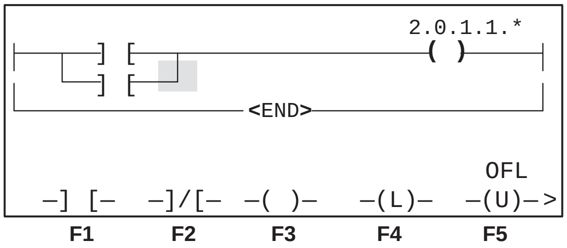


Inserting an Instruction

6. Press **`[ESC]`**, then **`[F1] – INS_INST`**, then **`[F1] – BIT`**, then **`[F1] –]   [–`**.

7. Type the bit address **`I:1/1`**, then **`[ENTER]`**.

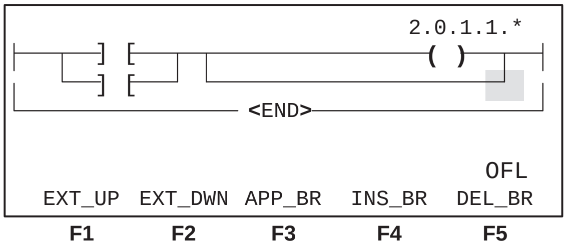
8. Then press **`[F5] – ACCEPT`**. The following display appears:



Adding an Output Branch

9. Press **`[ESC]`** twice. Then press the **[→]** cursor key once. Then press **`[F2] – BRANCH`**.

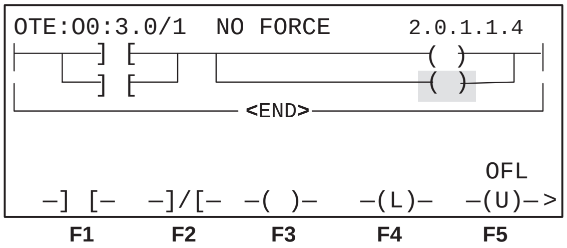
10. Press **`[F4] – INS_BR`**. Then press the **[→]** cursor key once, then **`[ENTER]`**. The following display appears:



Inserting an Instruction

11. Press **[ESC]** once, then **[F1] – INS_INST**, then **[F1] – BIT**, then **[F3] –( )–**.

12. Type the bit address **O:3/1**, then **[ENTER]**. Then press **[F5] – ACCEPT**. The following display appears:

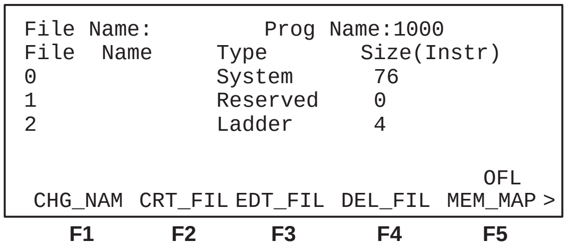


```
OTE:O0:3.0/1  NO FORCE      2.0.1.1.4
├──────] [──────────────────────( )────┤
      └─] [──┘  └───────────────( )──┘
└──────────────────<END>───────────────┘

                                   OFL
  —] [—   —]/[—  —( )—   —(L)—   —(U)—>
    F1      F2     F3      F4      F5
```

Saving the Program

13. Press **[ESC]** twice. Then press **[F5] – ACP_RNG**. Then press **[ENTER]**.

14. Press **[F5] – SAVE_EX**, then **[F5] – ACCEPT**. The following display appears:



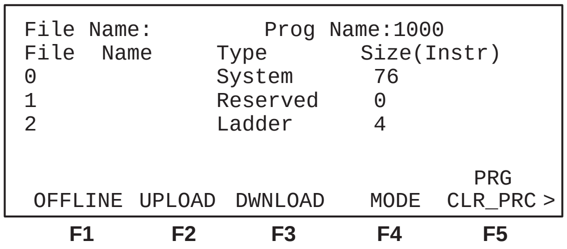
```
File Name:          Prog Name:1000
File  Name     Type        Size(Instr)
0              System       76
1              Reserved     0
2              Ladder       4


                                   OFL
 CHG_NAM  CRT_FIL EDT_FIL  DEL_FIL  MEM_MAP >
    F1       F2      F3       F4       F5
```

15. Press **[ESC]** once.

Downloading the Program

Start with the main display.

1. Press **[F5] – UTILITY**.
2. Press **[F2] – WHO**.
3. Press **[F3] – ATTACH**.
4. Put processor into PROGRAM mode: press **[F4] – MODE**, then **[F5] –PROGRAM**, then **[F2] – YES**. Then press **[ESC]**.
5. Press **[F3] – DWNLOAD**, then **[F2] – YES**. The following display appears:



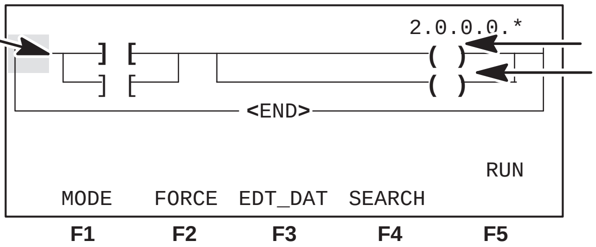
```
File Name:          Prog Name:1000
File  Name     Type        Size(Instr)
0              System       76
1              Reserved     0
2              Ladder       4


                                   PRG
 OFFLINE  UPLOAD  DWNLOAD    MODE   CLR_PRC >
    F1       F2      F3       F4       F5
```

6. Monitor the ladder program and put the processor into the RUN mode. To do this, press **`[ENTER]`**, then **`[F5] – MONITOR`**, then press **`2`**, then **`[ENTER]`**.
7. Press **`[F1] – MODE`**, then **`[F1] – RUN`**, then **`[F2] – YES`**, then **`[ESC]`**.

Testing the Ladder Program

1. Press pushbutton #0. Outputs #0 and #1 turn ON. The following display appears.

This input instruction becomes bold.

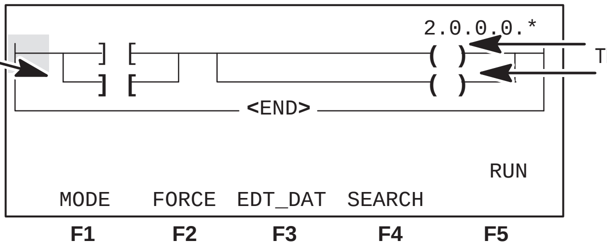


```
                              2.0.0.0.*
                 <END>
                                    RUN
  MODE    FORCE  EDT_DAT  SEARCH
```

These output instructions become bold.

2. Release pushbutton #0. Outputs #0 and #1 turn OFF.
3. Press pushbutton #1. Outputs #0 and #1 turn ON. The following display appears.

This input instruction becomes bold.



```
                              2.0.0.0.*
                 <END>
                                    RUN
  MODE    FORCE  EDT_DAT  SEARCH
```

These output instructions become bold.

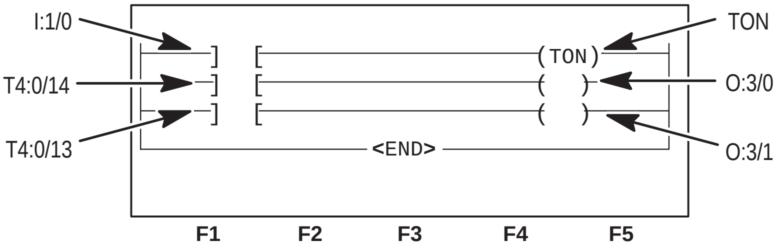
4. Release pushbutton #1. Outputs #0 and #1 turn OFF

To return to the main display, press **`[ESC]`** and respond **`YES`** to any prompts until the main display appears.

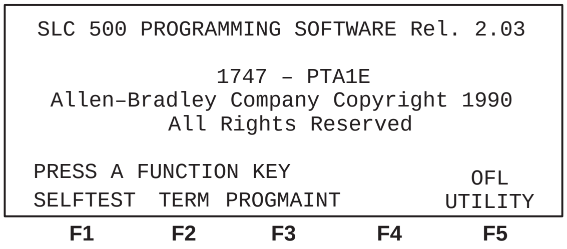
Entering a Timer Instruction

In this exercise, you enter a timer instruction with a time delay of 10 seconds. Two different types of timer status bits activate output pilot lights #0 and #1. The first type, called a "timer timing" status bit turns on output #0 for 10 seconds. The second type, called a "done" status bit, turns on output #1 *after* 10 seconds.

The program that you will enter in this exercise will look like the one in the display below:



Start with the main display below:

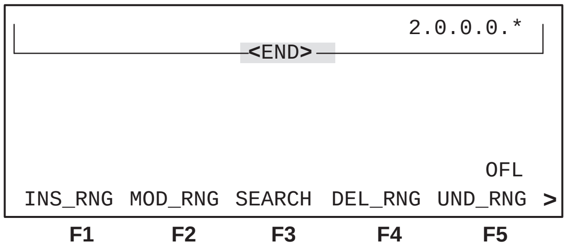


```
   SLC 500 PROGRAMMING SOFTWARE Rel. 2.03

                1747 – PTA1E
    Allen–Bradley Company Copyright 1990
             All Rights Reserved

  PRESS A FUNCTION KEY                  OFL
  SELFTEST  TERM  PROGMAINT          UTILITY
```

1. **[F5] – UTILITY**
2. **[F5] – CLR_MEM**
3. **[F2] – YES**
4. **[ESC]**
5. Rename program as shown in chapter 3, Creating a Program.
6. Reconfigure the controller as shown in chapter 3, Creating a Program.

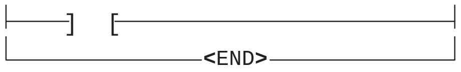
Entering the Program

1. Press **[F3] – PROGMAINT**, then **[F3] – EDT_FIL**.
2. Enter file number **2**, then press **[ENTER]**. The following display appears:



```
                                 2.0.0.0.*
                  <END>




                                       OFL
 INS_RNG  MOD_RNG  SEARCH  DEL_RNG  UND_RNG  >
```

3. Enter a rung and an XIC instruction. Press or enter the following:



A. `[F1] – INS_RNG`

B. `[F1] – INS_INST`

C. `[F1] – BIT`

D. `[F1] –] [–`

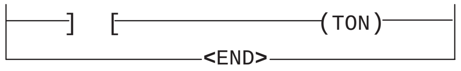
E. `I:1/0`

F. `[ENTER]`

G. `[F5] – ACCEPT`

H. `[ESC]`

4. Enter the Timer Instruction. Press or enter the following:



A. `[F2] – TMR/CNT`

B. `[F1] – TON`

C. `T4:0` (This is the Timer Address.)

D. `[ENTER]`

E. `1000` (This is the Timer Preset Value in hundredths of a second.)

F. `[ENTER]`

G. `0` (This is the Timer Accumulated Value.)

H. `[ENTER]`

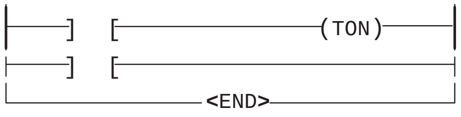
I. `[F5] – ACCEPT`

J. `[ESC]`

K. `[ESC]`

L. `[F5] – ACP_RNG`

5. Enter a second rung and an XIC instruction. Press or enter the following:



A. `[F1] – INS_RNG`

B. `[F1] – INS_INST`

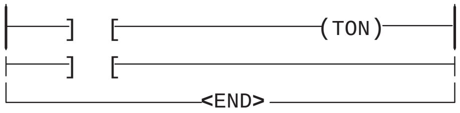
C. `[F1] – BIT`

D. `[F1] –] [–`

E. `T4:0/14` (14 represents the timer timing bit.)

F. `[ENTER]`

G. `[F5] – ACCEPT`



6. Enter an OTE instruction. Press or enter the following:

   A. **[F3] –( )–**

   B. **O:3/0**

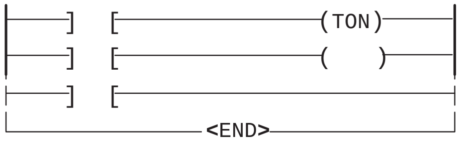
   C. **[ENTER]**

   D. **[F5] – ACCEPT**

   E. **[ESC]**

   F. **[ESC]**

   G. **[F5] – ACP_RNG**



7. Enter a third rung and an XIC instruction. Press or enter the following:

   A. **[F1] – INS_RNG**

   B. **[F1] – INS_INST**

   C. **[F1] – BIT**

   D. **[F1] –] [–**

   E. **T4:0/13** (13 represents the done bit.)

   F. **[ENTER]**

   G. **[F5] – ACCEPT**

8. Enter an OTE instruction. Press or enter the following:

   A. **[F3] –( )–**

   B. **O:3/1**

   C. **[ENTER]**

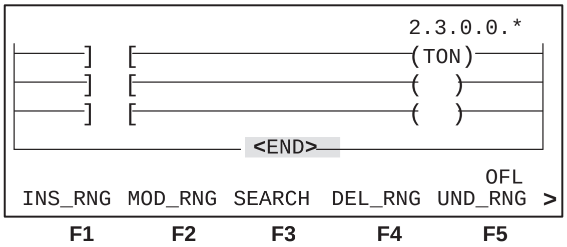
   D. **[F5] – ACCEPT**

   E. **[ESC]**

   F. **[ESC]**

   G. **[F5] – ACP_RNG**

Your final display should look like this:



9. Save the program. Press or enter the following:

A. `[ENTER]`

B. `[F5] – SAVE_EX`

C. `[F5] – ACCEPT`

D. `[ESC]`

10. Go online and run the program. Press or enter the following:

A. `[F5] – UTILITY`

B. `[F2] – WHO`

C. `[F3] – ATTACH`

D. `[F4] – MODE`

E. `[F5] – PROGRAM`

F. `[F2] – YES`

G. `[ESC]`

H. `[DWNLOAD]`

I. `[F2] – YES`

11. Monitor the ladder program. Press or enter the following:

A. `[ENTER]`

B. `[F5] – MONITOR`

C. `2`

D. `[ENTER]`

12. Change Processor Mode to Run. Press or enter the following:

A. `[F1] – MODE`

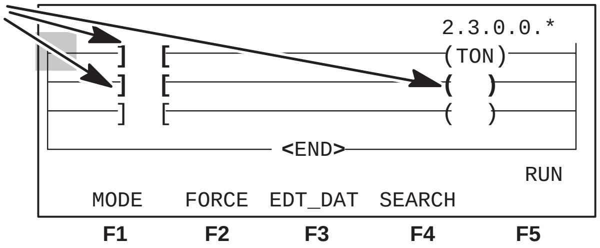
B. `[F1] – RUN`

C. `[F2] – YES`

D. `[ESC]`

13. Test your Ladder Program. Do the following:

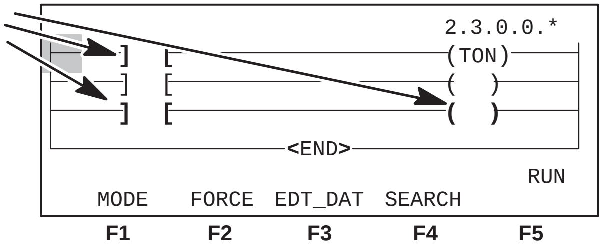
A. Press pushbutton #0 for at least 10 seconds. During the first 10 seconds, output #0 turns ON and #1 stays OFF. The following display appears:



```
                                      2.3.0.0.*
  ──] [─────────────────────────────(TON)───
  ──] [─────────────────────────────(   )───
  ──] [─────────────────────────────(   )───
  ──────────────────<END>───────────────────
                                          RUN
   MODE    FORCE  EDT_DAT  SEARCH
    F1      F2      F3       F4       F5
```

These bit instructions become bold.

B. After 10 seconds, output #0 turns OFF and output #1 turns ON. The following screen appears:

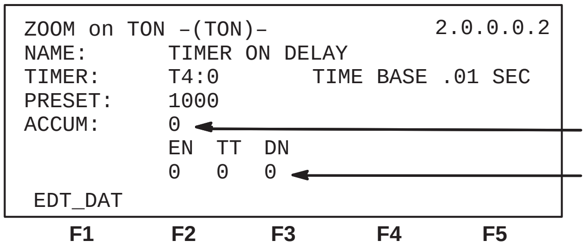


```
                                      2.3.0.0.*
  ──] [─────────────────────────────(TON)───
  ──] [─────────────────────────────(   )───
  ──] [─────────────────────────────(   )───
  ──────────────────<END>───────────────────
                                          RUN
   MODE    FORCE  EDT_DAT  SEARCH
    F1      F2      F3       F4       F5
```

These bit instructions become bold.

C. Release pushbutton #0; the timer resets and both outputs #0 and #1 turn OFF.

D. Place the cursor on the **TON** instruction (press the **[→]** key twice). Press **[ZOOM]**. The following screen appears:



```
ZOOM on TON –(TON)–                2.0.0.0.2
NAME:       TIMER ON DELAY
TIMER:      T4:0       TIME BASE .01 SEC
PRESET:     1000
ACCUM:      0
            EN  TT  DN
            0   0   0
 EDT_DAT
    F1      F2      F3       F4       F5
```

The accumulator value changes In step (e).

Notice that the values for EN, TT, and DN also change.

E. Repeat steps A, B and C. Watch the timer accumulator value change.

Troubleshooting Errors

This appendix shows you how to identify and correct errors that you may encounter while working through this guide. They include:

- HHT error messages
- System LED status
- Processor error codes

HHT Error Messages

The table below details error messages.

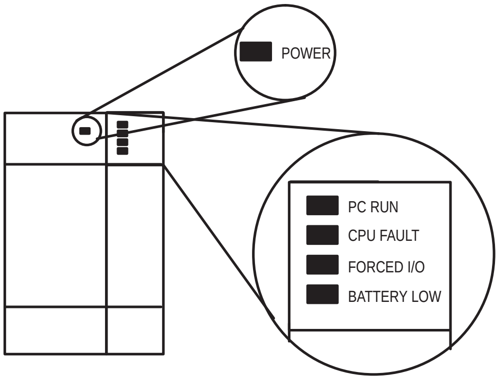
This Error Message:	Appears when:	Correct the Error by:
Change Processor to Program Mode	You attempt to download a program from the HHT to a processor that is in RUN mode.	Entering the MODE menu to change the processor from the RUN to PROGRAM mode.
Default File in Processor	You clear the processor memory, which is then automatically loaded with a “Default” program.	Downloading a program from the HHT memory in order to monitor controller operation.
Download Denied, Compiler Errors	You attempt to download a program saved with errors (possibly I/O configuration errors).	Going offline and checking the ladder program for any discrepancies between the I/O configuration and addresses used in the program. Either edit the program and change the address to agree with the configured I/O modules, or re–configure the I/O to match the entered address. For the latter, refer to chapter 3 for extra help. **Important:** You can SAVE the program with errors (to correct at a later time), but you cannot download the program to the processor.
Error: Undefined I/O Address	A mismatch exists between the I/O addresses used in the ladder program and the configured I/O modules.	Either editing the program and changing the address to agree with the configured I/O modules, or re–configuring the I/O to match the entered address. For the latter, refer to chapter 3 for extra help. **Important:** You can SAVE the program with errors (to correct at a later time), but you cannot download the program to the processor.
Illegal Operand	An address that you entered is not in the correct format.	Refering to chapter 2, Control Basics, for proper addressing format.
Illegal Network	You have a release of the HHT memory pak (<2.0) that does not support DH–485 network communications.	Obtaining an upgraded memory pak (release 2.0 or greater) from your local Allen–Bradley Sales Office or Distributor.
Incompatible Processor Type	The processor that you have configured in your program does not match the processor your HHT is communicating with.	Going offline and changing the processor type in the Processor Configuration. For extra help, see chapter 3.

This Error Message:	Appears when:	Correct the Error by:
Not a Processor	Either you are trying to ATTACH the HHT to either itself or a non–processor device while in the WHO utility.	Using the [↓] or [↑] keys to change the order of the nodes listed on the WHO screen. Put the processor at the top of the list and try to re–ATTACH.
	Or you are trying to ATTACH the HHT to a non–existent device, or no devices are shown on the WHO screen.	Changing the communication parameters of the HHT in the node configuration menu. From the WHO screen, press **`[F4] – NODE_CFG.`** Try changing the baud rate by pressing **`[F3] BAUD`**, the node address by pressing **`[F1] CHG_ADR`**, or the maximum node address by pressing **`[F2] MAX_ADR`**. Try different combinations. (The processor defaults at node address 1 and baud rate 19200.)
Program Files Differ	The ladder program in the processor memory does not match the program stored in the HHT memory.	Making the programs the same by either uploading the processor program into the HHT (overwriting the HHT memory) or downloading the HHT program to the processor (overwriting the processor memory). **Caution:** The overwritten program is lost.
Warning: Undefined I/O Referenced	The address you entered does not match the I/O configuration.	Either changing the address to agree with the configured I/O modules or exiting the edit mode and re–configuring the I/O to match the entered address. For the latter, refer to chapter 3, Creating a Program, for extra help.

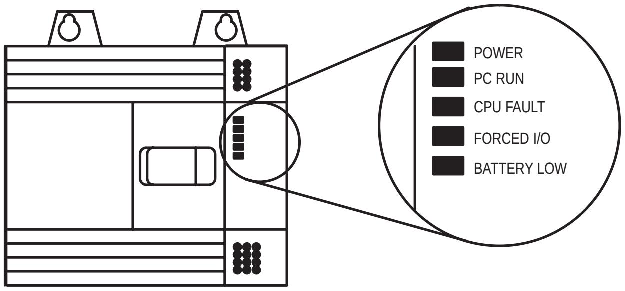
System LED Status

The System LEDs are located at different places on the modular system and the SLC fixed controller. Refer to the Installation and Operation manual for more information on system LED status. See the figure and table below:

Modular System



SLC 500 Fixed Controller



Processor LED	Status of LED	Cause	Corrective Action
POWER	Off	No DC power to backplane	Verify line power within range. Verify wiring and grounding. Check power supply fuse. For modular system, verify power supply jumper is positioned correctly and that the power supply has been sized properly for loading in the rack.
CPU FAULT	Steady	Bad CPU or firmware improperly installed	Verify that position of notch in processor firmware chip matches processor board diagram. See corrective action for Power LED.
CPU FAULT	Flashing	Processor major fault	Correct the problem based on the error message. First clear the fault. Then press and return processor to RUN mode. See processor error codes.
BATTERY LOW	Steady	Battery voltage fallen below threshold to retain RAM memory; or no battery present; or LED jumper is missing	Replace 1747–BA and do not remove processor power; or add 1747–BA to retain RAM memory; or add LED jumper to prevent the LED from lighting.

Processor Error Codes

The table below details some of the processor error codes. Refer to the HHT manual for a complete list of error codes and troubleshooting information.

Error Code	Cause	Corrective Action
0001	RAM program is corrupt due to noise, lightning, improper grounding, or loss of capacitor or battery back–up.	Check wiring, layout, and grounding. If using a 4K CPU, verify that a battery is installed to retain RAM memory when power is removed. See CPU FAULT–Flashing under system LED status. Download the program using the HHT.
0012	RAM program is corrupt or RAM itself is bad due to noise, lightning, improper grounding, or loss of capacitor or battery back–up.	Check wiring, layout, grounding. If using a 4K CPU, verify that a battery is installed to retain RAM memory when power is removed. See CPU FAULT–flashing under system LED status. Download the program using HHT.
XX50, XX51, XX52 XX53, XX54, XX55 (xx = slot #)	I/O module configuration/conflict or runtime problem.	Verify that processor configuration matches your hardware. See chapter 3 on configuring for keystrokes. Also see CPU FAULT — Flashing — under System LED status.
0056	Rack configuration error.	Verify that rack configuration in your program matches your hardware. See chapter 3 on configuring for keystrokes. If multiple rack system, verify proper installation of rack interconnect cable. See CPU FAULT — Flashing — under system LED status.

Identifying HHT Function Keys & Instruction Mnemonics

This appendix provides a listing of the abbreviated function keys and their meaning. It also provides a list of instruction mnemonics.

HHT Function Keys and Their Meaning

Below is a list of abbreviated function keys and their meaning.

Abbreviation:	Meaning:
ACCUM	accumulator value
ACP_RNG	accept rung
ADDR	address
APP_BR	append branch
B	battery
CAN_ED	cancel edit
CAN_RNG	cancel rung
CHG_NAM	change name
CLR_MEM	clear memory
CLR_PRC	clear processor
CONT	continuous
CPT/MTH	compute/math
CRT_DT	create data
CRT_FIL	create file
CSN	continuous scan
CUR_INS	current instruction
CUR_OPD	current operand
DEL_BR	delete branch
DEL_DT	delete data
DEL_FIL	delete file
DEL_INS	delete instruction
DEL_RNG	delete rung
DEL_SLT	delete slot
DWNLOAD	download
EDT_DAT	edit data
EDT_FIL	edit file
EDT_I/O	edit I/O

Abbreviation:	Meaning:
ENT	enter
ENT_MAS	enter master
EXEC_FILE	executable files
EXT_DWN	extend down
EXT_UP	extend up
F	force
FLT	fault
INS_BR	insert branch
INS_INST	insert instruction
INS_RNG	insert rung
I/O_MSG	I/O message
MEM_MAP	memory map
MEM_PRC	memory module to processor
MEM_SIZ	memory size
MOD_INS	modify instruction
MOD_RCK	modify rack
MOD_RNG	modify rung
MOD_SLT	modify slot
MOR_CPT	more compute
MOV/LOG	move/logic
NEW_INS	new instruction
NEW_PRG	new program
NEXT_FL	next file
NEXT_PG	next page
OFL	offline
PASSWRD	password
PRC_MEM	processor to memory module
PREV_FL	previous file
PREV_PG	previous page
PRG	program
PRG_SIZE	program size
PROGMAINT	program maintenance
RLY	relay
REM	remove
REM_ALL	remove all

Abbreviation:	Meaning:
REM_MAS	remove master
SAVE_CT	save and continue
SAVE_EX	save and exit
SEL_PRO	select processor
SFT/SEQ	shift/sequencer
SNK	sink
SRC	source
SSN	single scan
TMR/CNT	timer/counter
TRANS	transistor
UND_INST	undelete instruction
UND_RNG	undelete rung
WTCHDOG	watchdog
XFERMEM	transfer memory

Instruction Mnemonics

The table below provides a complete list of instruction mnemonics.

Mnemonic:	Instruction:
ADD	add
AND	and
BSL	bit shift left
BSR	bit shift right
CLR	clear
COP	copy file
CTD	count down
CTU	count up
DCD	decode 4 to 1 of 16
DDV	double divide
DIV	divide
EQU	equal
FFL	FIFO load
FFU	FIFO unload
FLL	fill file
FRD	convert from BCD
GEQ	greater than or equal to

Mnemonic:	Instruction:
GRT	greater than
HSC	high–speed counter
IID	I/O interrupt disable
IIE	I/O interrupt enable
IIM	immediate input with mask
INT	I/O Interrupt
IOM	immediate output with mask
JMP	jump to label
JSR	jump to subroutine
LBL	label
LEQ	less than or equal to
LES	less than
LFL	LIFO load
LFU	LIFO unload
LIM	limit test
MCR	master control reset
MEQ	masked comparison for equal
MOV	move
MUL	multiply
MVM	masked move
NEG	negate
NEQ	not equal
NOT	not
OR	or
OSR	one shot rising
OTE	output energize
OTL	output latch
OTU	output unlatch
PID	proportional integral derivative
REF	I/O refresh
RES	reset
RET	return from subroutine
RPI	reset pending I/O interrupt
RTO	retentive on–delay timer
SBR	subroutine

Mnemonic:	Instruction:
SCL	scale data
SQC	sequencer compare
SQL	sequencer load
SQO	sequencer output
SQR	square root
STD	STI disable
STE	STI enable
STS	STI start immediately
SUB	subtract
SUS	suspend
SVC	service communications
TND	temporary end
TOD	convert to BCD
TOF	timer off–delay
TON	timer on–delay
XIC	examine if closed
XIO	examine if open
XOR	exclusive or

Glossary

The following terms are used throughout this manual. Refer to them while working in this manual.

address: A character string that uniquely identifies a memory location. For example, I:1/0 is the memory address for the data located in the Input file location 1/0.

APS: (Advanced Programming Software) Software for a computer used to monitor and develop SLC 500 ladder logic programs.

attach: To establish communication with a processor.

bit: The smallest storage location in memory that contains either a 1 (ON) or a 0 (OFF).

branch: A parallel logic path within a rung of a ladder program.

communication scan: A part of the SLC's operating cycle. Communication with other devices, such as the HHT, takes place.

controller: A device, such as a programmable controller, used to monitor input devices and control output devices.

CMOS (Complementary Metal–Oxide Semiconductor): A semiconductor device in which an electric field controls the conductance of a channel under a metal electrode called a gate.

CPU (Central Processing Unit): The decision–making and data storage section of a programmable controller.

data file: An area within a processor file that contains the status of inputs, outputs, the processor, timers, counters, and so on.

download: To transfer a program from HHT memory to an SLC.

edit: To create or modify a ladder program.

expansion rack: A 2–slot rack used only with fixed controllers.

false: The status of an instruction that does not provide a continuous logical path on a ladder rung.

file: A collection of information organized into one group.

fixed controller: A controller with a power supply, CPU, and I/O integrated into a single package.

function keys: Keys on a personal computer or HHT keyboard labeled F1, F2 and so on. The operation of each of these keys is defined by HHT.

hardware: The components that comprise a programmable controller, such as Input modules, Output modules, and the CPU.

HHT: Hand–held Terminal.

I/O (Inputs and Outputs): Consists of input and output devices which provide and/or receive data from the programmable controller.

input device: A device, such as a pushbutton or a switch, that supplies data through input circuits to a programmable controller.

input scan: A part of the SLC's operating cycle. Status of the input modules are loaded into the Input data file.

instruction: A mnemonic and data address defining an operation to be performed by the processor. A rung in a program consists of a set of input and output instructions. The input instructions are evaluated by the SLC as being true or false. In turn, the SLC sets the output instructions to true or false.

ladder logic: A program written in a format resembling a ladder–like diagram. The program is used by a programmable controller to control devices.

memory: An area in either an HHT or processor that stores the ladder program (progam and data files).

memory pak: A circuit board that goes into the back of the HHT containing the software and language to operate the HHT.

mnemonic: A three–letter abbreviation for a ladder programming instruction.

modular controller: SLC 500 system consisting of a power supply, racks, CPU, and input and output modules.

module: An interchangeable plug–in device that may be inserted into a rack.

network: A series of devices connected by a communication medium.

offline: Describes devices not under direct communication. For example, when programming with an HHT.

online: Describes devices under direct communication. For example, when the HHT is monitoring the program file in a SLC. Also, a function in the HHT to establish communications with an SLC after using the WHO screen.

operating cycle: The sequential order of operations performed by the processor when in the run mode.

OTE (OuTput Energize): An instruction that energizes when a rung is true and de–energizes when a rung is false.

output device: A device, such as a pilot light or a motor starter coil, that receives data from the programmable controller.

output scan: A part of the SLC's operating cycle. During this scan the output data file information is transferred to the output modules.

processor: See CPU.

processor file: The set of Program and Data Files used by the SLC to control output devices. Only one processor file may be stored in the SLC at a time. In HHT terminology *program* is equivalent to *processor file*.

processor overhead: An internal portion of the operating cycle used for house–keeping and set–up purposes.

program: The set of Program and Data Files used by the SLC to control output devices. Only one processor file may be stored in the SLC at a time. In HHT terminology *program* is equivalent to *processor file*.

program file: The area within a processor file that contains the ladder logic program.

program mode: When the SLC is not executing the processor file and all outputs are de–energized.

program scan: A part of the SLC's operating cycle. During the scan the ladder program is executed and the Output data file is updated based on the program and the Input data file.

rack (chassis): A hardware assembly that houses devices such as I/O modules, processor modules, and power supplies.

read: To acquire data from a storage place. For example, the processor READs information from the input data file to solve the ladder program.

restore: To download (transfer) a program from the HHT to an SLC.

run mode: When the processor file in the SLC is being executed, inputs are read, the program is scanned, and outputs are energized and de–energized.

rung: Ladder logic is comprised of a set of rungs. A rung contains input and output instructions. During Run mode, the inputs on a rung are evaluated to be true or false. If a path of true logic exists, the outputs are made true. If all paths are false, the outputs are made false.

SLC (Small Logic Controller): A controller that comes in one of two styles: fixed or modular.

slot: The area in a rack that a module plugs into.

software: Executable programming package used to develop SLC ladder diagrams.

status: The condition of a circuit or system, represented as logic 0 (OFF) or 1 (ON).

terminal: A point on an I/O module that external I/O devices, such as a pushbutton or pilot light, are wired to.

true: The status of an instruction that provides a continuous logical path on a ladder rung.

upload: To transfer a program from an SLC to the HHT memory.

WHO: A utility in the HHT that lets you view the devices on your DH–485 network.

write: To copy data to a storage device. For example, the processor WRITEs the information from the output data file to the output modules.

eXamine If Closed (XIC): An input instruction that is logically true when the status of the bit located at its address is a 1; false when it's a 0.

eXamine If Open (XIO): An input instruction that is logically true when the status of the bit located at its address is a 0; false when it's a 1.

A-B QUALITY ALLEN-BRADLEY
A ROCKWELL INTERNATIONAL COMPANY

A subsidiary of Rockwell International, one of the world’s largest technology companies, Allen-Bradley meets today’s automation challenges with over 85 years of practical plant floor experience. 11,000 employees throughout the world design, manufacture and apply a wide range of control and automation products and supporting services to help our customers continuously improve quality, productivity and time to market. These products and services not only control individual machines, but also integrate the manufacturing process while providing access to vital plant floor data that can be used to support decision–making throughout the enterprise.

With offices in major cities worldwide.

WORLD HEADQUARTERS
Allen-Bradley
1201 South Second Street
Milwaukee, WI 53204 USA
Tel: (1) 414 382–2000
Telex: 43 11 016
FAX: (1) 414 382–4444

EUROPE/MIDDLE EAST/
AFRICA HEADQUARTERS
Allen–Bradley Europe B.V.
Amsterdamseweg 15
1422 AC Uithoorn
The Netherlands
Tel: (31) 2975/43500
Telex: (844) 18042
FAX: (31) 2975/60222

ASIA/PACIFIC HEADQUARTERS
Allen–Bradley
1201 South Second Street
Milwaukee, WI 53204 USA
Tel: (1) 414 382-2000
Telex: 43 11 016
FAX: (1) 414 382-2400

CANADA HEADQUARTERS
Allen–Bradley Canada Limited
135 Dundas Street
Cambridge, Ontario N1R 5X1
Canada
Tel: (1) 519 623–1810
FAX: (1) 519 623–8930

LATIN AMERICA
HEADQUARTERS
Allen-Bradley
1201 South Second Street
Milwaukee, WI 53204 USA
Tel: (1) 414 382–2000
Telex: 43 11 016
FAX: (1) 414 382–2400

1747–NM009 Series B—February, 1993
Supersedes 1747–NM009 Series A—May, 1992

40063–169–01(A)