GE Fanuc Automation

Programmable Control Products

Logicmaster™ 6 Programming and Documentation Software

User's Manual

GEK–25379D *May, 1990*

WARNINGS, CAUTIONS, AND NOTES AS USED IN THIS PUBLICATION

WARNING

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

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

CAUTION

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

NOTE

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

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

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

The Logicmaster™ 6 Programming and Documentation Software from GE Fanuc Automation North America, Inc. is used to create ladder logic programs for the Series Six™ programmable logic controller (PLC). This manual describes the Logicmaster 6 software package to be used with the Series Six PLC.

Using this Manual

Chapters 1 through 11 describe the features of the Logicmaster 6 software package. If you are starting up the system for the first time, or getting ready to do a printout, or displaying tables of data from the Series Six PLC, for example, you will refer to the chapter describing the function you want to use.

Chapters 12, 13, and 14 are for the programmer. Use them as a reference to the Series Six Instruction Set.

The manual is organized as follows:

Chapter 1. Introduction: provides an overview of the software package and options for running Logicmaster 6 software. Chapter 1 also describes the hardware requirements for using Logicmaster 6 software, plus the system's different modes of operation. A brief description of the principal features of the product is also provided.

Chapter 2. Operation: describes startup procedures for floppy-diskette and hard-disk users. The basic operation of the Workmaster computer, and other computers when using Logicmaster 6 software, is also described.

Chapter 3. Scratch Pad: The Scratch Pad function is used to display and store information about the current program in Logicmaster memory, or about the CPU capabilities. Chapter 3 defines the entries on the Scratch Pad display, and explains how to change these entries. A summary of the Scratch Pad function keys is also provided.

Chapter 4. Display Program: The Display Program function is used to display ladder logic, showing power flow through the rungs. Chapter 4 explains how to display a program, search for a program element, and make on-line changes. A summary of the Display Program function keys is also provided.

Chapter 5. Edit Program: The Edit Program function is used to create or modify a ladder logic program in Logicmaster memory. Chapter 5 explains how to use the software to enter and modify a ladder logic program: entering the Edit Program mode, editing the program, editing a rung, searching for a program element, and ladder diagram file editing. A summary of the Edit Program function keys is also provided.

Chapter 6. Annotation: Annotation can be used to add explanatory text to a ladder logic program. Chapter 6 explains how to create, display, and print annotation.

Chapter 7. Display Reference Tables: The Display Reference Tables function is used to display the status of any group of references. Chapter 7 explains how to enter, use, and exit this function.

Chapter 8. Print Functions: Print functions can be used to print copies of ladder logic and annotation. Chapter 8 explains how to set up the printer, define the content of a printout, and print in Foreground or Background mode. A summary of the Print function keys is also provided.

Chapter 9. Load/Store/Verify: The Load/Store/Verify function is used to transfer programs and tables to/from Logicmaster memory, transfer configuration data to/from the CPU, compare data, and clear Logicmaster memory. Chapter 9 explains how to perform these functions. A summary of the Load/Store/Verify function keys is also provided.

Chapter 10. Expanded Functions Menu: The Expanded Functions menu is used to set up and display CPU configuration, Series 90-70 I/O configuration, Genius I/O fault reporting, and communications

setup. Refer to chapter 10 for information on these functions. A summary of the function keys is also provided.

Chapter 11. Utility Functions: Utilities functions are used for disk and file management. Chapter 11 explains how to perform these functions. A summary of the function keys is also provided.

Chapter 12. Programming: presents general programming concepts. Refer to this chapter for information on basic ladder diagram format and the elements of a ladder diagram.

Chapter 13. Advanced Functions: Chapter 13 should be used as a reference guide for the Advanced functions. Within this chapter, functions are grouped into sections that correspond to the function key assignments used during programming. Within a section, each function is explained separately. After a description of the way the function operates in a program, you will find step-by-step instructions for entering the function in a rung.

Chapter 14. Expanded Functions: Chapter 14 should be used as a reference guide for the Expanded and Expanded II functions. Within this chapter, functions are grouped into sections that correspond to the function key assignments used during programming. Within a section, each function is explained separately. After a description of the way the function operates in a program, you will find step-by-step instructions for entering the function in a rung.

Appendix A. Setup Information: describes setup information for the Logicmaster 6 software.

Appendix B. Glossary of Terms: provides definitions of pertinent terminology.

Appendix C. Keyboard Translator Chart: provides a handy tear-out reference chart for use with IBM-PC computers plus a listing of the Alternate and Control key functions.

Appendix D. Reference Used Tables: indicate reference conflicts. The listings in this appendix show the characters that may appear in the Reference Used tables.

Appendix E. Software function Key Flow Diagrams: provides a map of the software function keys.

Appendix F. Error Codes: lists common error codes and their possible causes.

Appendix G. Annotation Files: describes annotation file structures for the Logicmaster 6 software. This information is provided as an aid to programmers creating customized documentation.

Related Publications

- GEK-25363 *Data Processor Unit User's Manual*
- GEK-25364 *Series Six™ Data Communications User's Manual*
- GEK-25366 *Redundant Processor User's Manual*
- GEK-25373 *Workmaster™ Guide to Operation*
- GEK-25398 *ASCII/BASIC Module User's Manual*
- GEK-83515 *I/O Transmitter Module Data Sheet*
- GEK-83524 *Interrupt Input Module*
- GEK-90527 *Cimstar™ I Industrial Computer Reference Manual*
- GEK-90771 *Advanced I/O Receiver Module Data Sheet*
- GEK-90802 *ProLoop Process Controllers System Manual*
- GEK-90824 *I/O Communication Control Module Data Sheet*
- GEK-90825 *I/O Link Local Module User's Manual*
- GEK-90846 *Genius I/O System User's Manual*
- GEK-96602 *Series Six™ Plus User's Manual*
- GEK-96669 *Alarm Master User's Manual*
- GFK-0013 *GEnet Factory LAN Network Interface User's Manual*
- GFK-0152 *Series 90-70 I/O Application Guide*

Linda McCoy
Technical Writer

GEK-25379

GEK-25379

GEK-25379

Chapter 1 Introduction

GEK-25379

Logicmaster™ 6 Programming and Documentation Software is used to create ladder logic programs for the Series Six family of programmable logic controllers (PLCs). Program development may be done on a Workmaster® or Cimstar™ I industrial computer, or an IBM PC-XT®, PC-AT®, Personal System 2® or compatible personal computer.

Both the Workmaster and Cimstar I computers are industrial-hardened, and recommended for installations where programs must be transferred, monitored, or edited in the harsh conditions of the factory floor. The Workmaster computer has the additional important advantage of easy portability.

After a program is developed, it is simple to transfer it to the CPU of a Series Six PLC. Then the Logicmaster 6 system can be used on-line with one or more operating CPUs, to provide continuously-updated displays of reference tables and program logic. The logic display features symbolic power flow through the rungs, so program execution can be traced.

Programming Features

Logicmaster 6 software offers a full range of programming functions, such as:

- Basic contacts, coils, timers, counters.
- Unsigned binary, signed single-precision, double-precision, and floating point arithmetic.
- Data Move, Table Move, List operations, and Matrix functions.
- Up to 16 subroutines in a single program. Other control functions such as Master Control Relay and Skip.
- Support for up to 16K CPU registers.
- Choice of conventional I/O tables or support for the Genius I/O system.
- Extensive, easy-to-display Help files.
- Printout of display screens, programs, annotation, and tables.
- Program storage on diskettes or hard disk.
- The ability to combine part or all of one ladder logic program with another.
- Up to 64K program memory.
- User-defined cross-reference tables.
- Powerful user-defined configuration instructions.

SECTION 1
System Configuration

Logicmaster 6 software is available in two versions, for systems using either parallel or serial communications with the Series Six PLC. Both versions are available on 3.5 and 5.25-inch diskettes.

Factory Floor Programmer

The parallel version on 3.5-inch diskettes includes a Factory-Floor Programmer diskette, which combines these functions on one diskette:

- Display Program.
- Edit Program.
- Display Reference Tables.
- Parallel Communication with the Series Six PLC programmable controller.
- Scratch Pad.
- Load/Store/Verify.
- Expanded Functions.

NOTE

The Factory Floor Programmer is not available on high density 3.5-inch media.

The contents of the diskette can be loaded into memory, allowing the diskette to be removed. The drive can then be used for program diskettes. *Note that maximum program size for the Factory Floor Programmer is 32K.*

Parallel and Serial Versions

While programs are under development, the parallel and serial versions are functionally similar. With a completed program transferred to a Series Six CPU, both versions can be used to monitor program execution, and communicate certain operator changes to the program. The parallel version provides greater response speed in situations where timing is critical. The serial versions allow connection over longer distances, and to more CPUs.

Both the parallel and serial versions of Logicmaster 6 feature:

- Operation on a system having any combination of one to four floppy-diskette drives, and/or one or two hard disk drives.
- Operation with or without the Extended Memory card.
- Backward-compatibility with the Release 2 and 3 versions of the Logicmaster 6 software.
- Support of the Enhanced Graphics Adapter (EGA) card.

Using a Workmaster or Workmaster II Computer

To run either the parallel or serial version of Logicmaster 6 software, the Workmaster or Workmaster II computer must have the following:

- 640K total available programmer memory (RAM).
- The correct version of DOS. For more information about DOS, refer to chapter 2, *Operation.*

It is advisable for additional printed circuit boards in the computer to be removed. For example, if the Workmaster computer has been used as part of a VuMaster™ system, the graphics board should be removed. Refer to GEK-25373, *Workmaster Guide to Operation* for instructions before you disassemble the computer.

Parallel Version

The parallel version can communicate with the Series Six PLC via a standard interface cable to the I/O system, as described in appendix A, *Setup Information.* Two printed circuit boards must be present in the Workmaster computer for parallel communications with the Series Six PLC.

- A Workmaster/Series Six PLC Interface Board.
- A Workmaster/Series Six PLC Terminator Board.

Serial Version

The serial version can communicate with the Series Six PLC over a serial communications channel to a CCM card. Communication is possible over a long distance, using a wide range of baud rates, with or without modems. The system can communicate with a single CCM card and CPU, or can be used in a multi-drop configuration having up to eight CCM cards and CPUs.

For point-to-point RS-232 communications over distances less than 50 feet, connection is made to the serial port on the Combination Adapter card in the Workmaster computer. This card can also be used for communications using modems. For RS-422 multidrop communications or point-to-point communications over distances greater than 50 feet, connection must be made to the serial port on the optional Asynchronous Joystick card (IC640BGB311).

Connection may be made to either a CCM2 or CCM3 card. If a CCM3 card is used, it must be operating in CCM2 mode, and must be version CCM31C600CB517C or later.

Optional Configuration

The Workmaster computer may be equipped with an Expanded Memory Card for overlay loading. Refer to chapter 2, *Operation*, for information.

Using a Cimstar I Computer

The Cimstar I computer can run either the parallel or serial version of the Logicmaster 6 software. The serial version requires no additional equipment. The parallel version requires an additional board set, which is described below.

Both versions require GE-DOS version 1 (equivalent to MS-DOS 3.2). This is the version of DOS originally supplied with the computer. For a hard disk system, this DOS version should be copied to the hard disk, as described in chapter 2, *Operation.*

Parallel Version

The parallel version will communicate with the Series Six PLC via a standard interface cable to the I/O system, as described in appendix A, *Setup Information.* The following boards must be present in the computer for parallel communications with the Series Six PLC:

- A Workmaster/Series Six PLC Interface Board.
- A Workmaster/Series Six PLC Terminator Board.

Serial Version

The serial version will communicate with the Series Six PLC over a serial communications channel to a CCM card. Communication is possible over a long distance, using a wide range of baud rates, with or without modems. The system can communicate with a single CCM card and CPU, or can be used in a multi-drop configuration having up to eight CCM cards and CPUs. Refer to chapter 2, *Operation* and appendix A, *Setup Information*, for the hardware requirements for serial communications.

Optional Configuration

The computer may be equipped with an Expanded Memory Card for overlay loading. Refer to chapter 2, *Operation*, for information.

Using an IBM Personal Computer

For programming applications that do not require an industrial-hardened computer, the Logicmaster 6 software will run on an IBM PC, IBM PC-XT, IBM PC-AT, IBM Personal System/2, or IBM Compatible personal computer with the following characteristics:

- 640K RAM.
- The correct version of PC-DOS.
  - Version 2.1 or 3.1 for the IBM PC and PC-XT.
  - Version 3.1 or 3.2 for the IBM PC-AT.
  - Version 3.3 or 4.0 for the IBM Personal System/2® computer.
- A color or monochrome monitor adapter card. The software will also support the Enhanced Graphics Adapter (EGA) card.

Parallel communications between Logicmaster 6 software and the PLC require a special board set (see appendix A). This board set occupies one full-size XT slot and an adjacent half-size slot.

Performance of the software with other versions of DOS is not guaranteed. Neither is performance guaranteed on other types of IBM compatible computers.

Parallel Version

The parallel version will communicate with the Series Six PLC via a standard interface cable to the I/O system, as described in appendix A, *Setup Information.* The following boards must be present for parallel communications with the Series Six PLC:

- A Workmaster/Series Six PLC Interface Board.
- A Workmaster/Series Six PLC Terminator Board.

Serial Version

The serial version will communicate with the Series Six PLC over a serial communications channel to a CCM card. Communication is possible over a long distance, using a wide range of baud rates, with or without modems. The system can communicate with a single CCM card and CPU, or can be used in a multi-drop configuration having up to eight CCM cards and CPUs. Refer to chapter 2, *Operation* and appendix A, *Setup Information*, for the hardware requirements for serial communications.

Optional Configuration

The computer may be equipped with an Expanded Memory Card for overlay loading. Refer to chapter 2, *Operation*, for information. If an Expanded Memory (RAM Disk) card is used for overlay loading, the overlay files must be copied to the RAM Disk before starting up the Logicmaster 6 software.

The system supports the IBM Monochrome Adapter Board and the Asynchronous Communications Adapter Board. It does not support serial communications adapters not based on the 8250 UART.

SECTION 2
Operating Modes

The system has three operating modes:

Off-Line Off-Line mode is used for program development. Power flow display or register contents are not updated from the CPU when the system is in Off-line mode.

On-Line On-Line mode provides real-time displays and program changes.

Monitor Monitor mode allows programs to be examined and real-time status displayed, but no changes of logic, register content, or I/O overrides are allowed.

In On-Line mode, the CPU periodically sends an updated input and output status table, register memory, override table, and Scratch Pad to the Logicmaster 6 system.

In On-Line mode, single word changes can be made to the program that is currently running in the CPU. These changes include changing a normally-open contact to a normally-closed contact, changing a relay coil to a one-shot coil, or overriding a contact. For more information, refer to chapter 4, *Display Program*, and chapter 5, *Edit Program*. In Monitor mode, the system can read data from the CPU, but may not transfer data to the CPU. All registers and tables are updated automatically to reflect the current operating state of the CPU. For the Workmaster computer, this is the only operating mode that allows removal of the key from the unit's keyswitch.

In Off-Line mode, tables and registers are updated in the Logicmaster 6 system memory (not the CPU) only upon command from the keyboard. Programs may conveniently be developed in Off-Line mode, without being connected to a CPU.

The following chart summarizes data transfer capabilities of the three operating modes:

Mode	I/O Table Memories Register Memory		Ladder Program (Logic Memory)	
	To CPU	From CPU	To CPU	From CPU
On-Line	manual	auto.	manual	manual
Monitor	no	auto.	no	manual
Off-Line	manual	manual	manual	manual

Mode Selection

Both the Workmaster computer and the Cimstar I computer have a keyswitch, which is used to select the operating mode.

The IBM PC version starts up in Off-line mode. Because the computer lacks the keyswitch, mode selection must be made in software. This can be done in either one of two ways:

1. Going to the Scratch Pad display and typing in the mode desired, then pressing a function key. For instructions, refer to chapter 3, *Scratch Pad.*
2. Pressing the ALT and 1 keys simultaneously. Repeatedly pressing ALT-1 switches operating mode from Off-line to Monitor to On-line, then back to Off-line.

CPU Keyswitch

In addition to the mode selection performed using the Logicmaster 6 system, an additional safeguard is provided when the system is connected to an operating CPU.

Parallel Version

The upper keyswitch on the CPU must be in the STOP position to store I/O and register tables. The ladder diagram can be changed when the CPU keyswitch is in STOP or RUN position.

Serial Version

The upper keyswitch on the CPU must be in STOP position to store I/O tables, register tables, and the ladder diagram to the CPU.

SECTION 3
Guide to Using Logicmaster 6 Software

This section introduces many of the features of the Logicmaster 6 software. ***If you have not used Logicmaster 6 software before, you should read this section first.***

The individual topics covered in this section will guide you to other chapters in the book, where more detailed information is located.

Section 3 introduces these subjects:

- How to duplicate the master software, set up serial ports, clear CPU parity errors, and handle files.
- How to use the Scratch Pad function to set up programming parameters, to change the Series Six CPU ID, or to turn the CPU on or off.
- The types of program annotation you can include in a program.
- How to edit, copy, and combine ladder logic programs.
- How to display tables of register and I/O values.
- How the serial version is set up to communicate with the Series Six CPU.
- How to transfer programs between the computer and the Series Six CPU, and between the computer and disks.
- The format of ladder logic, and how to display a program.
- The Logicmaster 6 program instructions.
- The features of the Factory Floor Programmer.

Help Screens

The Logicmaster 6 software includes detailed Help screens. For more information about the feature you are using, just press the Help (F10) key on your keyboard. The Help screens are easy to get into or out of. Program data will not be lost if you press the Help key. You will find the Help screens to be a very useful feature of the Logicmaster 6 software.

The Help screens are always available when the diskette containing them is present in the computer. For a system with hard disk memory storage, the Help files can be loaded onto the hard disk.

For a dual floppy-diskette drive system, the program diskette is normally in one drive and the Help diskette in the other.

For a single floppy-diskette drive system, the program diskette must be removed from the drive and the Help diskette inserted in order to use the Help files.

Utilities

Utility functions are probably the first functions of the Logicmaster 6 system you will use.

Duplicating the Master Software

When you start up the system for the first time, you will use the original diskettes shipped from the factory. You should make copies of these diskettes. If you have a floppy-diskette system, copying the master diskettes will give you a set of diskettes for everyday use. If you have a hard disk system, copying the master diskettes will place the Logicmaster 6 software on the hard disk. Chapter 11, *Utilities*, explains how to copy your master diskettes.

Setting up the Serial Ports

If you are using a serial version of Logicmaster 6, you will use the Serial Port Set Up utility. This utility is used to select the characteristics for serial ports. You must do this for the system to be able to communicate with the Series Six CPU. If you have a serial printer or other device, you will use the Serial Port Set Up utility before using the printer. Refer to chapter 11, *Utility Functions*, for instructions.

Clearing Parity Errors

Parity is a type of integrity check performed on memory areas in the CPU. If you are starting up a new CPU, or if a major hardware module has been added or changed, any ''parity errors'' should be cleared. This is done with the Utilities, as explained in chapter 11.

Using the File Utilities

You will use the File Utilities often. These are a group of DOS file-handling programs built into the Logicmaster 6 software. They are easier to use than conventional DOS programs, because they are ''menu-driven''. That means you do not have to remember and type DOS commands. Instead, you select the file utilities from a menu, and complete fully-prompted screens. For example, to copy a file you would select COPY FILE from the utilities menu, then complete this screen:

```
COPY FROM : SERIAL PORT/DRIVE ID      (1/A,B)
            FILE NAME
COPY TO   : SERIAL PORT/DRIVE ID      (1/A,B)
            FILE NAME
```

Information on copying, deleting, and renaming files is found in chapter 11.

Scratch Pad

The Logicmaster 6 system can easily be used to create programs off-line, perhaps in a location far from the Series Six PLC programmable controller. With Logicmaster 6 software, one computer can be used to create programs for many Series Six PLCs. The programs can be stored on diskettes, or a hard disk, and used whenever and wherever they are needed.

The Logicmaster 6 software can be used to create programs with the features described in this book. Your PLC may not be able to use all of these features. For instance, it may have less than 16K memory, or it may not have the Expanded function set. Or you may be creating programs for several PLCs that have CPUs with different capabilities.

A function called "Scratch Pad" is used to match the programming features of the Logicmaster 6 software to the CPU. When you select Scratch Pad (from a menu) the Scratch Pad Display appears. Here, you can select the features you want to include in the program for a specific CPU. For example:

```
CPU ID NUMBER :    005
MEMORY SIZE :      32K       WORDS USED :            2
SUBROUTINES USED : 00        WORDS AVAILABLE : 32,766
FUNCTIONS :  XPANDED/4.0     REGISTERS :          1024
```

In this example, the CPU which is identified as number 5 has 32K words of logic (program) memory. It has the Expanded function set. The Scratch Pad display also shows the number of words of memory, and the number of subroutines, that have been used by the program. In the example above, no program has yet been created. So far no subroutines have been used, and 32,766 words of logic memory are available for the program.

Changing the Scratch Pad

If you are going to create a program for a CPU with less memory or with a different CPU function set, first type in the information here. During programming, Logicmaster 6 software displays only the features selected in the Scratch Pad.

Using the Scratch Pad to Control the CPU

You can also use the Scratch Pad when the Logicmaster 6 system is on-line to the CPU, or in Monitor mode. Then, the information you see on the Scratch Pad comes from the CPU itself.

When the Logicmaster 6 system is on-line to a CPU, you can change the ID number in the CPU by changing the CPU ID NUMBER in the Scratch Pad. You can also turn the CPU scan on or off by simply pressing one key on your computer keyboard. Pressing another key controls the state of all hardware outputs if the CPU is running.

Chapter 3, *Scratch Pad*, explains how to use the Scratch Pad features.

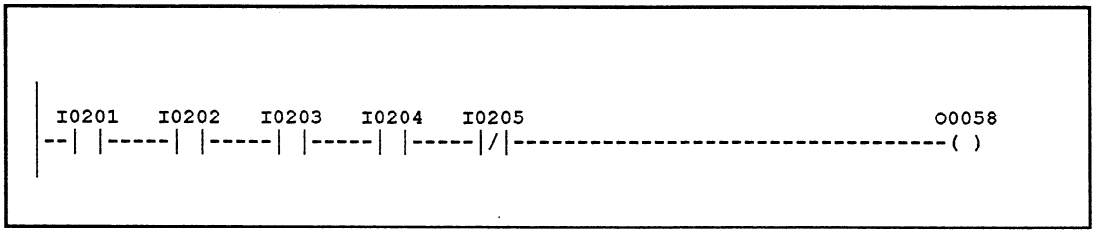
Annotation

Annotation is explanatory text in a program. This text makes a program easier to read and understand.

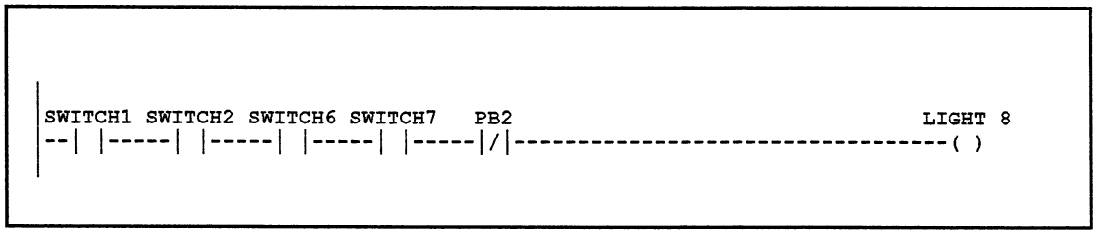
There are four basic types of annotation:

A ''nickname'' is another name for an input, an output, or a register. A nickname can have up to 7 characters. Examples of nicknames that might be used in a program are: LS035, PB11A, and SOL129C. When you display or print a program, you can see the nicknames above the program line.

This is an example of a program line without nicknames:

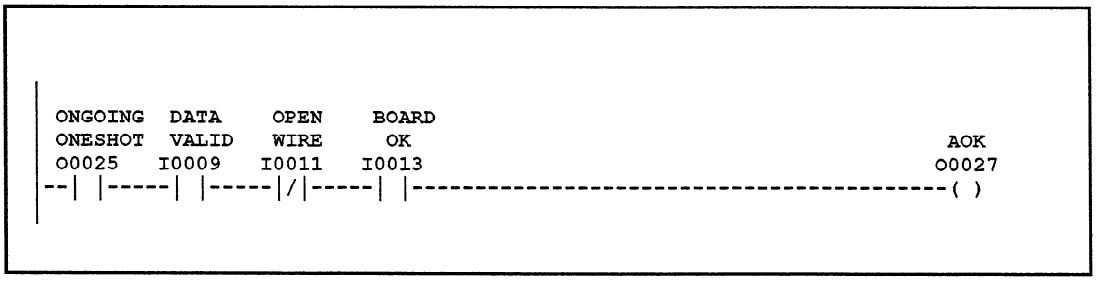


With nicknames for the inputs and outputs, the same line can be more meaningful when read.



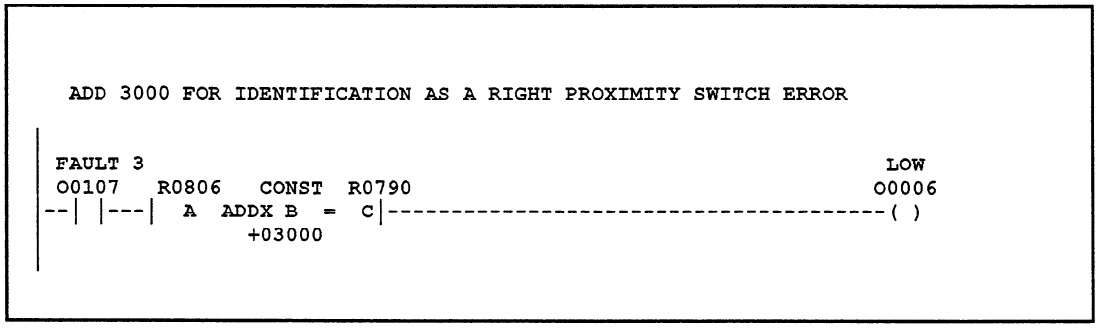
''Names'' are another kind of program annotation. Like nicknames, they can be placed above I/O and register references. A name for a program reference can consist of up to 21 characters. Examples of names are: HIGH LEVEL ALARM, and ERROR CONDITN DETECTD. You can also print or display names.

This example uses names (but not nicknames):



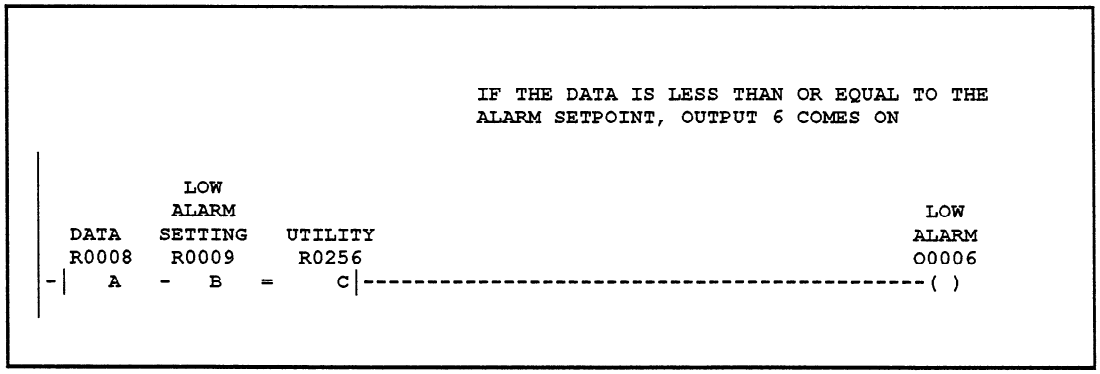
"Rung Explanations" between rungs explain the logic. Rung Explanations are easy to create or edit. They can contain any text you enter.

Example rung with a Rung Explanation:



The fourth kind of annotation is "Coil Labels". A coil label may appear either above or to the right of a coil. Like a rung explanation, a coil label is easy to create. It can contain any text you enter.

Example rung with a coil label:



Annotation can be displayed or printed. Special formats can be used to print borders around rung explanations and coil labels. Each page of the printout can be given a special title.

Refer to chapter 6, *Annotation*, for more information. For information about printing annotation see chapter 8, *Print Functions*.

Edit Program

When you create a new program or edit an existing program, you will select Edit Program from the Supervisor menu.

Edit Program mode provides a broad range of functions for editing ladder logic. Edit Program also includes text editing functions, for editing annotation. For information on editing annotation, refer to chapter 6, *Annotation*.

In Edit Program mode, the program is displayed graphically on the screen. To add a rung to a program, you select Insert Rung. This allows you to add elements to build the rung. The Insert Rung function provides access to the instructions for Series Six PLC programming. All of the program elements - from simple relays to floating point arithmetic - may be reached by selecting Insert Rung. You can limit the functions that can be used for a program. To do this, set up the Scratch Pad as described in chapter 3.

Each time you select a rung element, it appears on the screen at the current location of the cursor. For example, if you added a normally-closed contact to a rung, a graphic representation of a normally-closed contact would appear:

```
---|/|--
```

If you added an integer to Floating Point Conversion (one of the Expanded functions), the following would appear in the rung:

```
 *******          *******
-| INTEGER  TO  FLOATING POINT |-
```

Elements are added, with the appropriate values and references, to complete the rung. Explanations of all of the Logicmaster 6 program instructions are featured in chapters 12 through 14.

Copying and Combining Programs

Similar logic may be needed in more than one program. With Logicmaster 6, it is simple to copy part or all or one program into another program. This is done by combining program files, as explained in Chapter 4. Naturally, you must check to be sure the new program you have created makes sense.

It is also easy to create an editable copy of a program, leaving the original version unchanged. All you have to do is load the program you want to copy, and give it a new name. See chapter 5 for information.

Display Reference Tables

The Logicmaster 6 system maintains a set of tables which show the values of the inputs, outputs, and registers used in a program. To display a reference table, first type in the designation of a reference from that table. For example, type in an input (such as I0001) to display the Input table. Then, select Display Reference Tables from the Supervisor menu. The screen will display the table that includes the reference you entered.

If you selected the Input table, the first three lines of values in the Input table would look like this:

```
POINT #                    INPUT      0173          (nickname)
0064   00000000 00000000 10010100 00000000 10111111 00010010 01001001 01010100
0128   10101010 01001010 11010100 10101001 10101001 10010010 01010100 01010101
0192   11110101 11101000 10101000 00000010 00000100 01010111 11110111 01100101
```

The entire table shows 16 lines of input values, like the three example lines illustrated above. 1024 values are shown on the same page.

Outputs and register values are shown similarly.

If the Logicmaster 6 system is connected to a CPU and in On-Line mode, the values shown are from the CPU. Otherwise, they are from the computer's memory.

It is easy to change the numeric base for some or all of the items on the I/O displays or register displays. For instance, the I/O values can be in binary:

```
0064   00000000 00000000 10010100 00000000 10111111 00010010 01001001 01010100
```

Or converted to decimal. Here just 16 bits are converted:

```
                                      DECIMAL
                                    |_________|
0064   00000000 00000000 10010100          0191 00010010 01001001 01010100
```

Or to hexadecimal. Here the same 16 bits are converted to hexadecimal:

```
                                    HEXADECIMAL
                                    |_________|
0064   00000000 00000000 10010100          00BF 00010010 01001001 01010100
```

Register values can be converted similarly.

In addition to being able to display values, I/O references can be overridden and their status toggled on/off from the reference tables displays.

Chapter 7, *Display Reference Tables*, explains the types of reference tables. It tells how to display and how to use them.

Communicating with the CPU

The Logicmaster 6 system can transfer programs to and from the Series Six CPU, monitor program operation, and display tables of program data. To do this, it must be set up properly to communicate with the CPU.

Appendix A, *Setup Information*, describes how to connect the computer to the CPU.

Serial Communication

The serial version of Logicmaster 6 communicates with the Series Six CPU through a serial link to the CCM card.

For the serial version only, the Logicmaster 6 software includes a screen called the Communications Setup menu. On this screen, you enter the port number, the type of protocol, and the identifying number of the CPU.

```
COMMUNICATION PORT NUMBER:    (1,2)
COMMUNICATION PROTOCOL:    MASTER/SLAVE
SELECTED CPU ID NUMBER:    (1-90)
```

After this screen is completed, the Logicmaster 6 system can communicate with the selected CPU.

For information on the Communication Setup menu, refer to chapter 10, *Expanded Functions.*

Load/Store/Verify

While programs are being worked on, they are contained in the RAM memory of the computer. RAM memory can contain only one ladder logic program at a time. To be saved, a program must be "stored" on diskettes or a hard disk.

You will use the Load/Store/Verify functions to store programs. After selecting Store Program from a menu, all you have to do is specify where the program is to go, and type in its name. For example:

```
DRIVE ID  /  CPU  B  (A,B/P)  P = CPU
PROGRAM NAME    program1
```

Then, press the Shift and Enter keys. In this example, the program named PROGRAM1 will be stored on drive B. Drive B is the second diskette drive. It is often used for program diskettes.

If you look again at the example, you'll see that the Store Program feature is also used to send a program to the programmable controller CPU.

Loading a Program

"Loading" a program is the opposite of storing it. When you load a program, you transfer it into the computer's RAM memory. In RAM, the program can be viewed or changed.

In the example above, the program named PROGRAM1 was stored on a diskette in drive B. After that, the computer was turned off. To work on PROGRAM1 again, you would start up the computer, and load the program files that had been stored on the diskette.

After selecting Load Program from a menu, you would specify where the program was stored, and type in its name. For example:

```
DRIVE ID  /  CPU  B  (A,B/P)  P = CPU
PROGRAM NAME    program1
```

This time, when you press the Shift and Enter keys the computer reads the files for PROGRAM1 from the diskette into RAM.

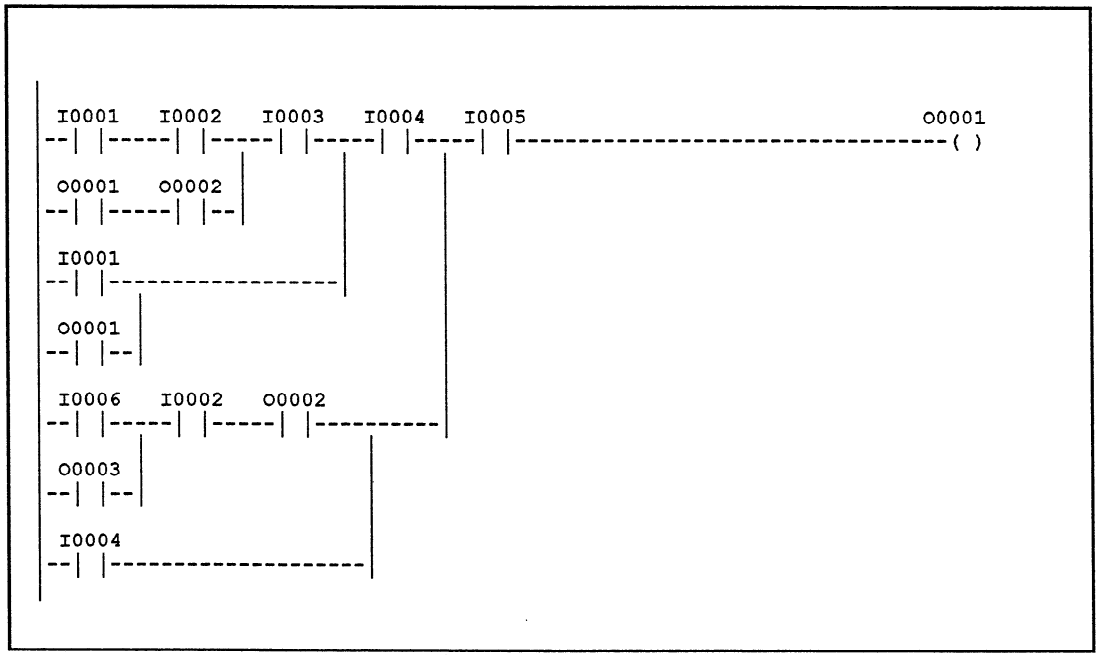
To be sure that a program has been either loaded or stored accurately, another function can be used to "verify" its content.

Chapter 9, *Load/Store/Verify*, explains how to use these functions.

Display Program

You can "display" a program in the computer's RAM memory. It can be a new program, or one you have placed in RAM using the Load function.

After entering the name of the program, all you have to do is select Display Program from a menu. The program will appear on the computer screen.



The display shows up to seven lines of logic at a time. You can quickly display any portion of the program using the cursor keys, or by ''searching'' for it.

On the display, you can see the symbolic flow of power through the rungs of the ladder logic.

With the computer in On-Line or Monitor mode, you can display the same program that is currently running in the Series Six CPU. The current values of the program elements will be displayed. The program display will include I/O states, overrides, and register contents. The Logicmaster 6 system allows certain changes to the program currently in the CPU. As described in chapter 4, *Display Program*, the types of changes that can be made include changing the base of the references on the display, changing the element type (for example normally-open contact changed to normally-closed), or overriding inputs and outputs.

Chapter 5, *Edit Program*, explains how to make more extensive changes to a program.

Program Instruction Sets

There are three function sets available with the Logicmaster 6 software: Advanced, Expanded and Expanded II. These correspond to the function sets present in logic modules for the Series Six PLC programmable controller. The Logicmaster 6 software can be used to create programs for a Series Six PLC with any of these instruction sets. The instructions used in the program should be matched to the function level of the CPU for which the program is intended.

The functions available in each function set are listed below.

Advanced Functions

The Advanced functions provide all the basic elements for a ladder diagram: contacts, coils, unsigned binary arithmetic, and other program functions. They also include the use of auxiliary I/O references, signed double-precision arithmetic, subroutines, table and matrix functions, and the other functions listed below. For more information about these functions, refer to chapter 13, *Advanced Functions*.

Group	Function
Relay	Normally Open and Normally Closed Contacts Real Outputs Internal Outputs Latches One-Shot Coils Timers (tenths, hundredths, seconds) Counters (up and down) Auxiliary I/O References
Arithmetic	Unsigned Binary Addition Unsigned Binary Subtraction Unsigned Binary Compare Shift Double Precision Addition Double Precision Subtraction Double Precision Division Double Precision Multiplication Greater Than

Group	Function
Control	Master Control Relay Skip Do Subroutine Do I/O Suspend I/O Return Status
Move/Convert	I/O to Register Data Move Register to I/O Data Move BCD to Binary Conversion Binary to BCD Conversion Table to Destination Move Source to Table Move Move Table Move Table Extended Move Right 8 Bits Move Left 8 Bits Block Move Move A to B
Communications Requests	Data Processing Request (DPREQ) Serial Communications Request (SCREQ)
Matrix	AND Inclusive and Exclusive OR Invert Masked Compare Set/Sense Bit Clear/Sense Bit Shift Bit Right Shift Bit Left
List	Add to Top Remove from Bottom Remove from Top Sort
Miscellaneous	End of Sweep No Op

Expanded Functions

The Expanded functions include all the Advanced functions. In addition, the Expanded functions include the use of 16K of I/O points, floating point arithmetic, and enhanced Do I/O, direct addressing of full 16K of registers, enhanced Status, and enhanced Data Processing Request functions. The Expanded II function set includes one function, 90-70 I/O Rack Service, which is used to service a Series 90 -70 I/O rack.

The Expanded functions are listed below. For more information about these functions, refer to chapter 14, *Expanded Functions.*

Group	Function
Relay	16K I/O References All Basic and Advanced Relay Functions
Arithmetic	Floating Point Addition Floating Point Subtraction Floating Point Division Floating Point Multiplication Floating Point Compare Integer to Floating Point Conversion Floating Point to Integer Conversion Expanded Compare All Basic and Advanced Arithmetic Functions
Control	Do I/O Enhanced Status Enhanced All Basic and Advanced Control Functions CPU Configuration Function 90-70 I/O Rack Service Function
Move/Convert	All Basic and Advanced Move/Convert Functions
Communications Requests	Serial Communication Request - same as Basic and Advanced Data Processing Request, Enhanced
Matrix	All Basic and Advanced Functions
List	All Basic and Advanced Functions
Miscellaneous	Basic/Advanced Functions

Factory Floor Programmer

The parallel version of the Logicmaster 6 software on 3.5-inch diskettes includes an additional diskette called the Factory Floor Programmer. This diskette contains the principal software functions of the Logicmaster 6 software. It provides a quick debugging tool for the factory floor.

NOTE

The Factory Floor Programmer is not available on high density 3.5-inch media.

Use the Duplicate Master function on the system diskette to copy the original Factory Floor Programmer diskette for everyday operation.

To use the Factory Floor Programmer, start up the computer in DOS, as explained in chapter 2. Then load the contents of the Factory Floor Programmer diskette into memory. The diskette can then be removed if you want to use the same drive for program diskettes.

The Factory Floor Programmer provides the following functions:

Function	Description
Display Program	Display a program, make on-line changes, and search for ladder elements.
Edit Program	Edit a program with either an active or inactive file name, and search for ladder elements.
Display Reference Tables	Display all types of reference tables, including mixed reference tables. Change formats and values of all reference addresses.
Scratch Pad	Make changes to all entries on the Scratch Pad display, such as CPU ID and memory size, and function set.
Load/Store/Verify	Load a program from a disk or from the CPU, store a program, verify a program, and clear memory in the computer.
Expanded Functions	Modify and display CPU configuration. Also includes the Clear Parity function, used to clear CPU parity errors.
Teach	Provide the functions of Teach mode, as described in chapter 2, *Operation.*
Special Keys **Parallel Communications**	Provide ALT key functions listed in chapter 2, *Operation.* Perform all monitoring and communication functions of the parallel version.

The Factory Floor Programmer does not include: printing, annotation features, Help screens, file-handling and diskette utilities, port setup, or monitoring of Genius I/O faults. Programs edited or displayed with the Factory Floor Programmer are limited to 32K in size.

Chapter 2 Operation

GEK-25379

This chapter explains what you will need to know to start up the Logicmaster 6 system, and to use the features described in this book.

There are 8 sections in this chapter:

Section 1. Using DOS: describes DOS versions, and explains how to start up DOS, use DOS commands, format diskettes, and exit from Logicmaster 6 to DOS.

Section 2. Installing the Logicmaster 6 Software: Before you use the Logicmaster 6 software for the first time, follow the installation instructions in section 2.

Section 3. Starting up the Logicmaster 6 Software: describes how to start up the Logicmaster 6 software using diskettes or a hard disk.

Section 4. Using the Features of the Supervisor Menu: describes the entries on the Supervisor menu. This section also explains how to specify a file name from the Supervisor menu.

Section 5. Starting up the Series Six PLC: explains how to set up serial communications between the Series Six programmable controller and the Logicmaster 6 system, and also how to start up the Series Six PLC with the Logicmaster 6 software.

Section 6. Using Your Keyboard: explains the types of keyboards that may be used with the Logicmaster 6 software. This section also explains how to define special key assignments to make programming easier.

Section 7. Entering Data: explains the types of information you will see on the screen. This section also explains how you will enter data, such as file names and numerical values.

Section 8. Working with Numbers: provides information about number types: binary, decimal, BCD, hex, double-precision, and floating point.

SECTION 1
Using DOS

The acronym DOS stands for Disk Operating System. DOS is a software program that interfaces other programs (like Logicmaster 6) with the computer hardware. DOS must be used to start up the Logicmaster 6 software.

NOTE

For Logicmaster 6 software release 3.02 and later, DOS must be supplied separately.

As with other types of software, there are different versions of DOS. To run the Logicmaster 6 software, you must use a compatible version of DOS. For a Workmaster industrial computer or a Cimstar I industrial computer, you should use MS -DOS version 3.2.

If you have a copy of the earlier GE version 3.14 of MS™-DOS, it can be used for a Workmaster computer with diskette drives or a hard disk. For an IBM PC or PC-XT computer, you must use PC-DOS version 2.10 or later. For an IBM PC-AT, you must use PC-DOS version 3.0 or later. The term PC-DOS refers to the software supplied by IBM as "DOS Disk Operating System for IBM personal computers".

To determine what version DOS you have, start up the computer in DOS. The copyright screen will display the DOS version number. If you are already in DOS, you can display the DOS version by typing **VER** after the A> prompt.

Starting up DOS

DOS can be run from the diskette or from a hard disk. To start up DOS:

1. The DOS software must be in the computer. If DOS is not installed on a hard disk, place your DOS diskette in drive A.
2. Start up the computer by turning on the power to the computer. If the computer was already on, you can restart it by pressing CTRL-ALT-DEL, or a Reset button.
3. The copyright screen will appear while the DOS software is loaded into RAM memory. When the date and time prompts appear, enter the requested information and press the Return key to continue.
4. The DOS command prompt **A>** appears. This prompt shows the letter ID of the drive. If you used a diskette in drive A to start up DOS, then the prompt will indicate drive A. This is the "current" drive.
5. The cursor appears at the prompt. To use one of the DOS commands, you would type it here and press the Return key. To run a software program from a diskette or hard disk, you would enter its command line.

Changing the Drive ID

DOS assigns an identifying letter to each drive in the computer. The letter assignments depend on the type of computer you are using, and how its hardware is set up.

When you start up or reset the computer, it will check drive A for a diskette with certain system files. If these files are present, the contents of the diskette in drive A will be loaded into the working memory (RAM) area of the computer. This is what happens when you load DOS from a diskette. After the software is loaded, the drive A command prompt will appear. Drive A is now the current drive.

If the computer does not find the system files in drive A, it will look for a hard disk. (If a non-system disk is in drive A, an error message is displayed.) If the system files are on a hard disk, it will load those files into RAM and the hard disk drive prompt will appear. In this case, the hard disk becomes the current drive.

To change the current drive, enter the new drive ID followed by a colon and press the Return key. For example, to change the current drive from C to A, enter **A:** at the C> prompt and press the Return key. The current drive will now be displayed as **A>**. The computer will now go to drive A to execute commands, rather than to drive C.

DOS Commands

Your DOS manual explains how to use DOS, and provides complete definitions of all DOS commands. Each time you enter a command, DOS will perform some function for you. The following commands are used to install and start up Logicmaster 6 software:

DOS Command	Description
COPY	Copy one or more files, either on the same disk or from one disk to another. For example, to copy a file named CONFIG.WM to the same disk, and call the copy CONFIG.SYS, you would enter: COPY config.wmconfig.sys
REN	Rename a file. For example, to rename a file named CONFIG.WM as CONFIG.SYS on the same drive, you would enter: REN config.wmconfig.sys
TYPE	Display a file on the computer screen. For example: TYPE config.sys

Any kind of file can be displayed with the Type command. If the file is not a readable ASCII file, it will appear as an assortment of characters and beeps. One kind of file displayed in that way is a ladder logic file. You cannot use DOS to display a ladder logic file in recognizable form. To see the ladder logic, you must enter Logicmaster 6 software, load the file into RAM memory, and then use the Display or Edit Program function.

Formatting Diskettes

Before you can use a new diskette, it must be initialized. The DOS Format command prepares a diskette to receive data.

NOTE

Formatting erases any data previously placed on the diskette. Do not format a diskette that contains data you want to save.

To format a diskette:

1. Start up the computer in DOS. At the DOS prompt, type:

```
format (drive:)
```

2. Insert a write-enabled diskette into the specified drive, and press the Return key. Follow the prompts to format the diskette.

Finding a File

The DOS DIR command will display a listing of all the files on a drive. This listing is called a ''directory''. The directory shows the name of each file, its extension, its size, and the date and time it was last edited.

To display a directory, enter the DIR command.

- Entering **DIR** displays the directory of the disk in drive A, when drive A is the current drive.
- Entering **A> DIR B:** displays the directory of the disk in drive B, when drive A is the current drive.
- Entering **B> DIR** displays the directory of the disk in drive B, when drive B is the current drive.

The DIR command causes the file listing to scroll rapidly upward on the screen. If the listing is too long to fit on one screen, you can display just one screen at a time by entering the characters **/P** after the command. For example, entering **A> DIR B:/P** would cause DOS to display a directory of the files on the disk in drive B. After the first screen of the listing is displayed, press any key to display the next screen.

Wildcard characters can be used to display a directory of related files on a disk. For example, entering **A> DIR B: *.LAD** would display a directory of all .LAD files on the disk in drive B.

Exiting to DOS

When Logicmaster 6 software is running in the computer, you can exit directly to DOS from the Supervisor menu. You may want to do this, for example, if you need to format a diskette using DOS.

Be careful exiting to DOS if there is a ladder logic program in RAM memory that you want to keep. Exiting to DOS is just like restarting the computer; the contents of RAM memory will be lost. If you have been editing a program that has an active file name, it has automatically been saved to disk with each time you pressed CTRL-A (or the Accept key). If the program does not have an active file name, before exiting to DOS you should store the program using the Load/Store/Verify function of Logicmaster 6 software. If necessary, you can use any formatted diskette to store the program.

To exit to DOS from the Logicmaster 6 Supervisor menu:

1. Press ALT-Z. If your computer has a hard disk, it will return to DOS. If you are using DOS diskettes to start up the computer, your screen may display the following prompt:

   ```
   Insert disk with \COMMAND.COM in drive A
   and strike any key when ready
   ```

   Place your DOS diskette, containing the COMMAND.COM file, in drive A. Press any key to continue. If you do not have a diskette with a COMMAND.COM file, you cannot complete this procedure. Refer to your DOS manual for more information about the COMMAND.COM file.

   If you are running Logicmaster 6 software from diskettes, the screen may now prompt:

   ```
   Insert disk with batch file
   and press any key when ready
   ```

   If that prompt appears, place diskette 1 of the Logicmaster 6 diskettes, which contains a batch file, in drive A. Press any key to continue.

2. The DOS command prompt will appear. If you have been using diskettes, it will be the drive A prompt: **A>**.

3. To return to the Logicmaster 6 software, enter its command line: **LM6**. (If you are using diskettes, the Logicmaster 6 diskette 1 must be in drive A.) Then, press the Return key.

SECTION 2
Installing Logicmaster 6 Software

This section explains how to prepare new Logicmaster 6 software diskettes for everyday use. Ordinarily, you will use this information only once, to create startup diskettes or to install the Logicmaster 6 software on a hard disk. After the installation has been completed, you can follow the routine startup procedures given in section 3.

Before you use the Logicmaster 6 software regularly, you should follow the instructions in this section.

1. You should print out a copy of the README.LM6 file, as described in this section. The README file contains recent information about the software.
2. Check the content of your System Configuration file. This file must agree with your hardware setup. If necessary, edit or create this file on your startup disk.
3. If your computer has a hard disk, install the Logicmaster 6 software on the hard disk.

All files provided with Logicmaster 6 software are read-only files. They can be renamed, but they cannot be copied over, deleted or edited. Unused files can be left on the disk; they will not interfere with operation of the Logicmaster 6 software. To edit one of these original files (including any of the System Configuration files) you must copy and rename the file first, and then edit the copy.

Logicmaster 6 System Diskettes

Logicmaster 6 software is available for systems using either parallel or serial communications with the Series Six programmable controller. Both versions are available on either 3.5-inch or 5.25-inch diskettes, as described below.

3.5-inch diskettes are used for computers, such as the Workmaster or Cimstar I industrial computer, which use 3.5-inch diskettes with 80-track format. 5.25-inch diskettes are used for computers, such as the IBM PC-XT or PC-AT personal computers, which use 5.25-inch diskettes with 40-track format.

Diskette	Type of Diskette	Description
	3.5-Inch Diskettes	
Diskette 1	System diskette	The files on diskette 1 enable you to display the Supervisor menu. Following the instructions in this section, you can make diskette 1 into a startup diskette.
Diskette 2	Overlay diskette	The files on diskette 2 provide the features of the Supervisor main menu. These files are called "Overlay" files. Following the instructions in this section, you can load these files to another diskette, a hard disk, or RAM Disk for faster execution. Diskette 2 also contains Help files. These files contain the explanatory text for the Help screens.
Diskette 3	System diskette	This diskette, for the Factory Floor Programmer, is provided only with the parallel version of Logicmaster 6 software. The Factory Floor Programmer includes, on one diskette, an abbreviated set of Logicmaster 6 functions for use on the factory floor.

Diskette	Type of Diskette	Description
		5.25-Inch Diskettes
Diskette 1	System diskette	The files on diskette 1 enable you to display the Supervisor main menu.
Diskette 2	Overlay diskette	The files on diskette 2 provide the features of the Supervisor main menu. These files are called "Overlay" files. Following the instructions in this section, you can load these files to another diskette, a hard disk, or RAM Disk for faster execution.
Diskette 3	Overlay diskette	Diskette 3 contains the Print functions and Help files, which contain the explanatory text for the Help screens.

Reading/Printing the README File

Before using Logicmaster 6 software, read the file named README.LM6, which is provided on one of the diskettes. If possible, you should also print a copy of the README file and place it with this manual. The README.LM6 file contains the latest information about the Logicmaster 6 software.

To read or print the README file:

1. Start up the computer using DOS. You can use DOS that is installed on your hard disk, or place a DOS diskette in drive A of your computer. Cycle power to the computer. After the date and time function, the DOS prompt appears.
2. Place either diskette 1 (for the Workmaster or Cimstar I industrial computer) or diskette 3 (for the IBM personal computer) of the Logicmaster 6 software in a diskette drive.
3. To print the README file without reading it, go to step 4 below. To read the file, at the DOS prompt, type:

   ```
   type (drive:)readme.lm6
   ```

   Press the Return key. The file will appear, scrolling up the screen. To stop the scrolling, press the CTRL and S keys at the same time. To resume scrolling, press CTRL-S again. To stop displaying the file, press CTRL-C while the screen is not scrolling. The DOS prompt will reappear.
4. To print the README file, at the DOS prompt, type:

   ```
   print (drive:)readme.lm6/p
   ```

   Press the Return key. If this is the first time the Print command has been used since powerup, the screen prompts:

   ```
   Name of list device [PRN]:
   ```

   To send the file to the printer, press the Return key. Be sure that the printer is turned on, and in the On-Line mode. The printing will take place in background mode, and the DOS prompt will reappear.

To stop the printout before it is completed, at the DOS prompt type:

```
print(drive:)readme.lm6/c
```

Printing will continue until the end of the print buffer in the printer is reached. To end the printout more quickly, turn off the printer.

System Configuration File

To run Logicmaster 6 software, the disk you use to start up the computer must contain a System Configuration file with the name CONFIG.SYS. When a hard disk is used as the startup disk, a file named CONFIG.SYS must be in the root directory.

The System Configuration file is a short, readable file that describes the system requirements for the software. Different software packages use different System Configuration files, depending on the software requirements and the hardware configuration. The final content of this file will depend on the configuration of your system. For Workmaster and Cimstar I computers, versions of the System Configuration file are provided with the Logicmaster 6 software. For an IBM personal computer, you must supply a CONFIG.SYS file.

The content of the CONFIG.SYS file must be appropriate for your system. For a Workmaster or Cimstar I computer, a version of the file supplied with the Logicmaster 6 software can be copied, and used as is in many cases. After copying the file, check the content.

1. For all Logicmaster 6 applications, the file must contain these three lines:

   ```
   device = ansi.sys
   ```

   The entry for buffers must be 5, and the entry for files must be 20. The device driver calls for a file named ANSI.SYS. The ANSI.SYS file must be present on the DOS disk at startup for the Logicmaster 6 software to run.

2. For a Workmaster computer, the CONFIG.SYS file must also contain the following line, which enables the device driver for the Workmaster clock:

   ```
   device = wmclock.sys
   ```

3. If you are using a RAM Disk card from GE Fanuc - NA, your CONFIG.SYS file must contain one of the following pairs of lines:

   A. For a computer with diskette drives:

   ```
   device = gexmem2.sys-c21c
   device = gexdisk.sys-k640
   ```

B. For a computer with a hard disk:

```
device = \lm6\gexmem2.sys-c21c
device = \lm6\gexdisk.sys-k640
```

The values for these device drivers (-c21c and -k640) may not be correct if you have changed the RAM Disk's configuration register address (switch 2) or memory allocation (switches 7 and 8). The value (-k640) refers to the amount of expanded memory that you wish to reserve for use as a fast-access floppy-disk. You can reserve up to the total listed in appendix A. For more information, refer to appendix A, *Setup Information*, and to the instructions provided with your Expanded Memory Card.

4. If you are using the MS-DOS version 3.2 and your Workmaster computer has an external 5.25-inch diskette drive, you should place a DRIVPARM command in the CONFIG.SYS file. This command tells DOS the capacity of the diskettes placed in the drive, which is important when formatting diskettes.

   For example, if you have a Workmaster computer with two 3.5-inch drives (A and B) and an external 5.25-inch floppy-disk drive (C), the following line will specify that drive C has the standard 40-track 360K byte form factor:

   ```
   drivparm-/d:2 /f:0
   ```

   In the example, the 2 refers to drive C (0=A, 1=B, 2=C, etc.) and 0 refers to form factor 0 (360K). If you do not use this line, DOS assumes form factor 2 (760K), which is correct for 3.5-inch drives. Your DOS manual has more information on the parameters of the DRIVPARM command.

If the CONFIG.SYS file on your DOS disk does not contain the above lines, you must create a file that does. You can copy one of the files listed below, or modify your existing file. It is wise to save your existing CONFIG.SYS file in case you need to refer back to it. Then, you can copy one of the above files to make a CONFIG.SYS file.

If changes are still required to the CONFIG.SYS file, you will need to know how to use a text editor. A brief discussion of EDLIN (the line editor supplied with DOS) can be found later in this section.

Running Other Software with the CONFIG.SYS File for Logicmaster 6

Other types of software may require different entries in the CONFIG.SYS file. It is not always possible to combine the requirements for multiple software packages in one CONFIG.SYS file. In that case, you must maintain multiple versions of the CONFIG.SYS file. That can be done by using multiple DOS diskettes, or by renaming versions of the CONFIG.SYS file and using the one required for a specific application. Your DOS manual contains more information about the CONFIG.SYS file.

Versions of the System Configuration File

Versions of the sample System Configuration files provided with the Logicmaster 6 software are shown below. The list shows the content of each file, the diskette where it is located, and the type of computer for which it is intended.

Disk	File Name	File Content	Computer Type	RAM Disk	Hard Disk
diskette 1	CONFIG.WM	buffers=5 files=20 device=ansi.sys device=wmclock.sys device=gexmem2.sys -c21c device=gexmdisk.sys -k640	Workmaster	yes	no
	%GNRCFG.SYS	buffers=5 files=20 device=ansi.sys device=wmclock.sys	Workmaster	no	yes/ no
	%GRCFG.SYS	buffers=5 files=20 device=ansi.sys device=wmclock.sys device=\lm6\gexmem2.sys -c21c device=\lm6\gexmdisk.sys -k640	Workmaster	yes	yes
	CONFIG.PC	buffers=5 files=20 device=ansi.sys	Workmaster II Cimstar I IBM PC	no	no
*diskette 3 (parallel)	CONFIG.SYS	buffers=5 files=20 device=ansi.sys device=wmclock.sys	Workmaster Cimstar I	no	no
**MS-DOS diskette	CONFIG.SYS	break=on buffers=20 files=20 device=ansi.sys	Cimstar I	no	no
	CONFIG.WM	break=on buffers=5 files=20 device=ansi.sys device=wmclock.sys	Workmaster	no	no

*Diskette 3, the Factory Floor Programmer, is supplied only with the parallel version of Logicmaster 6 software.

**The latest GE version of MS-DOS 3.20 is supplied only for the Workmaster or Cimstar I computer.

Installation: Computer with Diskette Drives

If you are using a computer with diskette drives, follow these steps to install the Logicmaster 6 software:

- Duplicate the master software diskettes.
- Prepare a startup diskette.

After completing these steps, you can follow the startup procedures in section 3 of this chapter.

Duplicating the Master Software Diskettes

Begin by making copies of the original diskettes for everyday use, as described below. Keep the originals in a protected location.

1. Place the DOS diskette in drive A and start up the computer. After the date and time prompts, the DOS command prompt appears.
2. Format a diskette for use as a system diskette. At the DOS prompt, enter:

```
FORMAT (drive:)/s      (press the Return key)
```

3. Place another diskette in the computer. *(Formatting will destroy any data already on the diskette.)* Respond to the prompts to format the diskette.
4. When the Format utility is complete, remove the diskette and label it LM6 disk 1.
5. For the parallel version of Logicmaster 6 software, system diskette 3 contains the Factory Floor Programmer. If you want to duplicate this diskette, format another diskette as a system diskette and label it LM6 disk 3.
6. Place another diskette in the computer, and format it as a non-system diskette by entering:

```
FORMAT (drive:)      (press the Return key)
```

   Remove the newly-formatted diskette, and label it LM6 disk 2.
7. Place Logicmaster 6 system diskette 1 in drive A. Type either **LM6WM**, for a Workmaster or Cimstar I computer, or **LM6PC**, for an IBM PC or IBM-compatible computer. Then, press the Return key.
8. When prompted by the Logicmaster 6 title screen, press any key. Complete the Date and Time screen; then, press CTRL-E (or the Enter key) to display the Supervisor menu.
9. When the Supervisor menu appears, replace diskette 1 of the Logicmaster 6 software with diskette 2. In the Supervisor menu, select Utility Functions (F8).
10. From the Utilities menu, select Duplicate Master (F1). Use this utility to duplicate the contents of the master Logicmaster 6 diskettes onto the newly-formatted diskettes.

Preparing a Startup Diskette

If you want to use Logicmaster 6 diskette 1 for startup, follow the instructions below.

NOTE

The original master diskettes cannot be made "bootable". Prepare duplicate diskettes, as previously described, before starting this procedure.

To prepare Logicmaster 6 system diskettes that can be used for startup:

1. Place the DOS diskette in drive A and a duplicate Logicmaster 6 diskette 1 in drive B. Continue below at the correct step for the type of computer you are using:
   A. Workmaster (no RAM Disk):
      (1) Rename the CONFIG.WM file to CONFIG.SAV. To do this, enter **REN b:config.wmconfig.sav** and press the Return key.
      (2) Rename the %GNRCFG.SYS file to CONFIG.SYS by entering **REN b:%gnrcfg.sysconfig.sys**. Then, press the Return key.
   B. Workmaster with RAM Disk: On diskette 1, rename the CONFIG.WM file to CONFIG.SYS. To do this, enter **REN b:config.wmconfig.sys**. Then, press the Return key.
   C. Cimstar I or IBM PC: On diskette 1, rename the CONFIG.PC file to CONFIG.SYS. To do this, enter **REN b:config.wmconfig.sys** and press the Return key.
2. Copy the ANSI.SYS file from the DOS diskette to the Logicmaster 6 diskette by entering **COPY a:ansi.sys b:**. Then, press the Return key.
3. For a Workmaster computer only, copy the Workmaster clock file from the DOS diskette to the Logicmaster 6 diskette by entering **COPY a:wmclock.sys b:**. Then, press the Return key.
4. For the parallel version only, replace diskette 1 in drive B with diskette 3 (Factory Floor Programmer diskette). Repeat steps 3 and 4 above.
5. Now, either diskette 1 or 3 can be used to start up the computer. For startup instructions, refer to section 3 of this chapter.

Installation: Computer with a Hard Disk

Follow the instructions below to install Logicmaster 6 software on a computer with a hard disk. The hard disk should already be formatted, and DOS should be installed. To start up the Logicmaster 6 software, the hard disk must have an appropriate CONFIG.SYS file in the root directory. In addition, there must be an ANSI.SYS file on the hard disk.

If the hard disk has been used for a previous version of Logicmaster 6 software, check the root directory for an existing LM6PC.BAT or LM6WM.BAT file. If one is present, delete or rename it before using the Duplicate Master utility, as described in step 1 below.

1. This step is *only* required for those users who are installing new Logicmaster 6 software on a hard disk, where an earlier version of the software is already installed.

   To recover space on the hard disk, follow the steps described here before installing your new Logicmaster software on the hard disk. If this procedure is not followed, some old files will remain on the hard disk when the new software is installed. Other files related to previous versions of the software will be written over by the Duplicate Master utility.

   A. At the DOS prompt, with the hard disk selected as the current drive, go to the Logicmaster 5 subdirectory by entering **CD LM6**.

   B. Delete the previous versions of Logicmaster 6 software by entering **delete %G*.***. Then, press the Return key.

   C. Delete the old Logicmaster 6 batch file by entering **delete LM6.BAT**. Then, press the Return key.

2. Start up the computer in DOS. When the DOS command prompt appears, place Logicmaster 6 diskette 1 in drive A and enter either **LM6WM**, for a Workmaster or Cimstar I computer, or **LM6PC** for an IBM personal computer or an IBM-compatible computer. Then, press the Return key.

3. Answer the prompts until you get to the Supervisor menu. When the Supervisor menu appears, replace diskette 1 in drive A with diskette 2. Select Utility functions by pressing F8.

4. In the Utilities menu, select Duplicate Master (F1). Place Logicmaster 6 master diskette 1 (not a copy) into a diskette drive. At DUPLICATE FROM, enter the designation of the drive containing the master diskette. Move the cursor to DUPLICATE TO and enter the designation of the hard disk drive to receive the copy. Available drive letters are displayed beside the prompt.

5. Press the Enter key. If not already present, the utility creates the directory /LM6 on the hard disk. System files are then copied into the LM6 subdirectory. When the copying is finished, the screen displays the message "DUPLICATION COMPLETED."

6. Repeat the process to copy the other master diskette(s). Then, return to the Utilities menu by pressing F8.

7. When you installed the Logicmaster 6 software, you installed two System Configuration files that might be used by a computer with a hard disk. For a Workmaster or Cimstar I computer, you may wish to replace the CONFIG.SYS file already on the hard disk with one of these two files:

```
%GRCFG.SYS      if the computer uses a RAM Disk card.
%GNRCFG.SYS     if the computer has no RAM Disk card.
```

The content of these files was shown earlier in this section. If you want to use one of these as the CONFIG.SYS file, rename the present CONFIG.SYS file to save it. Then, copy one of the above files and name the copy CONFIG.SYS. The drive ID for either file is the ID of the hard disk.

8. If the CONFIG.SYS file on the hard disk needs to be changed, instructions are given later in this section. If the CONFIG.SYS file is changed, you must re-start the computer to utilize the changes. Instructions on the use of EDLIN and more information about editing the file are given in this section.

CONFIG.SYS File

Use the DOS Directory (DIR) command to display the contents of your DOS disk. If there is a CONFIG.SYS file on the disk, you can display its contents by entering **TYPE config.sys** and pressing the Return key.

If you do not have a CONFIG.SYS file, create one as described below. Note that this procedure will write over any existing version of the CONFIG.SYS file. If such a version exists, rename it CONFIG.BAK before following these steps:

1. At the DOS prompt, type **copy con config.sys**. Then, press the Enter key.
2. Type in the configuration commands needed. Press the Return key after each command. When you have finished typing the commands, press F6 and then the Return key to save the new CONFIG.SYS file.

Editing the CONFIG.SYS File

If you have a CONFIG.SYS file that does not include all the parameters needed to run Logicmaster 6 software, you can edit the file using a line editor, such as EDLIN. A simple procedure to edit the CONFIG.SYS FILE using EDLIN follows. Your DOS manual contains complete instructions for using the EDLIN commands and features.

In the procedure below, the example entries assume that the DOS diskette being edited is in drive A. If you need to end the use of EDLIN without saving the file, type **Q** at the asterisk prompt and press the Return key to return to the command prompt.

1. Enter the EDLIN command and the name of the file. For example, at the A>prompt, type **EDLIN config.sys** and press the Return key. EDLIN will prompt:

```
End of input file
```

2. The asterisk (*) is the EDLIN prompt. The first step in editing the file is to display the file contents. To do this, enter the List command by typing **L** at the asterisk. Then, press the Return key. The CONFIG.SYS file will be listed as individual numbered lines.

```
*L
  1: break=on
  2: buffers=5
  3: files=20
  4: device=ansi.sys
```

3. Compare the contents of your file with the entries required to run Logicmaster 6 software. You can insert, delete, or edit lines as described below. Note that these EDLIN commands can only be executed at the EDLIN asterisk prompt at the left side of the screen (not from a numbered line). To execute an EDLIN command, return to the asterisk prompt by pressing CTRL-C:

```
2: buffers=5
3: files=20
4: device=ansi.sys
5: *^C
```

A. To insert a line in the file, enter the Insert command at the EDLIN asterisk prompt by typing **I**. Then, press the Return key.

The sequence of items in the CONFIG.SYS file is not important. However, if you want to insert the line in a particular order, you can specify a line number before entering the Insert command. For example, to have the inserted line as line 5 in the file, you would type **5I** at the asterisk and press the Return key.

This line number will appear beside the asterisk prompt. Now, type in the content for the line. For example:

```
5:*drivparm=/d:0 /f:2
```

Press the Return key at the end of the line.

Continue until all the necessary lines have been added to the file. After entering the last new line, press the Return key again. The next line number appears:

```
5:*drivparm=/d:0 /f:2
6:*device=wmclock.sys
7:*
*
```

Press CTRL-C to return to the EDLIN asterisk prompt:

```
5:*drivparm=/d:0 /f:2
6:*device=wmclock.sys
7:*^C
```

B. To delete a line from the file, enter the Delete command at the asterisk prompt by typing **D** and the number of the line to delete. The next example removes the Workmaster clock device driver from the file.

```
5:*drivparm=/d:0 /f:2
6:*device=wmclock.sys
```

You should use the List command to verify the deletion.

C. To edit part of an existing line, enter its line number at the asterisk prompt. In the next example, line 2 is selected for editing.

```
1: break=on
2: buffers=20
3: files=20
4: device=ansi.sys
5:*drivparm=/d:0 /f:2
6:*device=wmclock.sys
```

The line appears again on the screen:

```
2: buffers=20
2:*
```

Enter the correct line and press the Return key.

```
2: buffers=20
2:* buffers=5
```

Press CTRL-C to return to the EDLIN asterisk prompt.

```
7:*^C
```

4. To finish using EDLIN, enter the End command by typing **E** at the asterisk. Then, press the Return key. The End command saves the new version of the file under the original file name (here, CONFIG.SYS). Also, it automatically creates a backup version of the file named CONFIG.BAK.
5. After you enter the End command and press the Return key, the DOS command is displayed. You can check the contents of the file by entering **TYPE config.sys**. Then, press the Return key.
6. After you create or edit a CONFIG.SYS file, restart the computer. This *must* be done in order to use the entries in the new file. If the computer is not restarted, the previous version of the CONFIG.SYS file that was present the last time the computer was started up will continue to be used.

SECTION 3
Starting up the Logicmaster 6 Software

After the installation procedures described in the preceding section have been completed, you can start up the Logicmaster 6 software normally.

If you are also using the computer to run other types of application software, 640K of RAM memory must be available for Logicmaster 6 software. If the message "Software Initialization Failure Number 2" is displayed when the software is started up, check for batch files that are loading other software into RAM memory at startup.

Follow these steps to start up the Logicmaster 6 software.

NOTE

If you are using a computer with a hard disk, Logicmaster 6 software should be installed on the hard disk, as previously described in this chapter. With the software installed on a hard disk, begin at step 3 below.

1. Start up the system using appropriate DOS, as explained in the preceding section. If the computer is turned off, apply power. If the computer is already turned on when you inserted the diskette, you can perform a software start by pressing the CTRL-ALT-DEL keys at the same time. After the DOS software is loaded, the DOS command prompt (A>) will be displayed.
2. Insert Logicmaster 6 diskette 1 in drive A.
3. At the DOS command (drive) prompt, type either **LM6WM (driveID)**, for a Workmaster or Cimstar I industrial computer, or **LM6PC (drive ID)**, for an IBM personal computer or IBM-compatible computer.

   The drive ID is optional. Enter a drive ID only if you want the computer to look for the overlay files on a different drive. See below for more information.

   If you specify the hard disk or RAM Disk, the files on the overlay diskette(s) must have been copied to that drive previously.
4. After entering the Logicmaster 6 command line, press the Return key. Follow the prompts to display the Supervisor menu.

Overlay Files

Overlay files are the files on diskette 2 of the Logicmaster 6 software. These files provide the functions of the Supervisor menu, for example, Edit Program or Scratch Pad. These files can be loaded to another drive for convenience, or for faster execution. The drive may be a diskette drive, hard disk, or the RAM Disk card.

For example, if you are using a Workmaster computer with dual diskette drives, you may place Logicmaster 6 diskette 1 in drive A and the Overlay diskette (diskette 2) in drive B. You would then enter the command **LM6WM B** when you start up the Logicmaster 6 software:

If the overlay files have already been loaded to a RAM Disk card, as explained below, enter the RAM Disk drive ID in the command line by typing **LM6WM C**. The RAM Disk is not always assigned the drive ID letter C. If you are not sure of the drive ID of the RAM Disk card, refer to the explanation that follows.

You can store the overlay files to a RAM Disk card (Expanded Memory card) or hard disk. The hard disk provides permanent storage for the files. If you store the files to a RAM Disk, they will only remain as long as power is present. When the system is turned off, files stored on the RAM Disk are lost. The overlay files must be loaded to the RAM Disk card each time the computer is powered up.

Using a RAM Disk for Overlay Files

If you will be using the RAM Disk to store the Logicmaster 6 overlay files, copy the overlay files to the RAM Disk before entering the startup command line (LM6WM or LM6PC). Logicmaster 6 software will **not** automatically copy the overlay files to the RAM Disk.

If this is the first time you have started up the computer with the RAM Disk drivers activated, determine the drive name DOS has assigned to the RAM Disk. DOS assigns the drive name depending on the numbers and types of disk drives in the system at the time the RAM Disk is set up by the instructions in the CONFIG.SYS file. The RAM Disk becomes the highest available drive.

For example, if you have dual diskette drivers on a Workmaster computer, then the RAM Disk becomes drive C. (Drive names A and B are always reserved for one or two floppy-diskette drives.) If you have a hard disk drive, then the RAM Disk may become drive D or drive G, depending on the version of DOS you are using.

If you are not sure where the RAM Disk has been assigned, you can locate the drive experimentally by using the DIR command. The RAM Disk is the last drive in the chain of drive names assigned by DOS. If you do a DIR on the RAM Disk, you should get either a list of files or the message "File not Found" because the RAM Disk is empty.

To test whether or not you have located the last drive in the chain, attempt to read the directory for a drive named with the next letter of the alphabet. For example, to test if the RAM Disk is drive G, enter the command **DIR H:**. The message "Invalid drive specification." should be displayed. If it is not, then drive G is not the last drive in the chain and it is not the RAM Disk.

Copying Overlay Files to the RAM Disk

Once you have determined the drive ID for the RAM Disk, you can copy the Logicmaster 6 overlay files to the RAM Disk using the DOS COPY command. The wildcard character '?' may be used to simplify this process.

For example, if your RAM Disk is drive G, you could place the overlay disk in drive A and type the command **COPY A:%g?lm6.00? G:**. This command will copy all eight of the overlay files for your Logicmaster 6 software to the RAM Disk.

Remember that the overlay files must be loaded to the RAM Disk *each* time you power up the computer. If you want this done automatically, you can put the COPY command into a batch file, such as AUTOEXEC.BAT. Simple instructions for creating an AUTOEXEC.BAT file follow. For more information about batch files, you should refer to your DOS manual.

1. If you are using a DOS disk to start up the Logicmaster 6 software, create the .BAT file on the DOS disk. If you are using DOS on a hard disk to start up the Logicmaster 6 software, create the .BAT file in the root directory of the hard disk. Do this only if you want the overlay files loaded to the RAM Disk every time you start up the computer.

2. Enter the following:

```
COPY CON AUTOEXEC.BAT
COPY A:%%g?lm6.00? G:
```

   The date and time lines are DOS commands. Including these in the file will cause the date and time prompts to appear at powerup.

   Notice that there are now two percent signs in the file name for the COPY command.

3. Press CTRL-Z to end the file; then, press the Return key.

The procedure above will load the Overlay files to the RAM Disk automatically. It is still necessary to specify the location of these files when you start up the Logicmaster 6 software. For example, if the RAM Disk is drive C of a Workmaster computer, you would type **LM6WM C**.

The overlay files will remain on the RAM Disk until you delete them, or until power is turned off. You can do a software start of the system by pressing CTRL-ALT-DEL without deleting the files, but using the Reset button will clear the RAM Disk.

Displaying the Title Screen

At power-up the title screen is displayed.

```
GE Fanuc Automation North America, Inc.

   LOGICMASTER (TM) 6 version 04.xx
Programming and Documentation Software
for SERIES SIX Programmable Controllers



Copyright General Electric Company 1984, 1985, 1986
  Copyright GE Fanuc Automation N.A., Inc, 1987
            ALL RIGHTS RESERVED

          Strike a key when ready ...
```

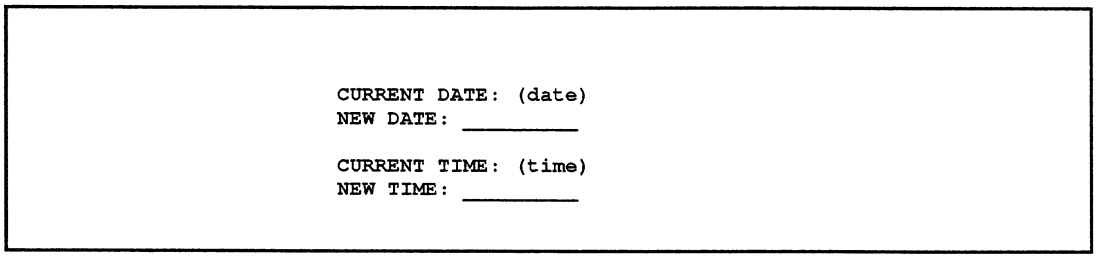
This screen identifies the version of the software being used. The title screen for the parallel version of Logicmaster 6 is shown above.

For serial versions of the software, at startup the system looks for a file named COMSET.SET. This file describes the serial communications configuration of the system. If the system is started up in Monitor or On-Line mode (implying communications with the CPU), the serial version attempts to establish communications with the CPU, and read in the Scratch Pad settings if the COMSET.SET file is present and specifies a valid port. If an invalid port is specified, the screen displays the message "INVALID PORT SPECIFIED IN COMSET.SET."

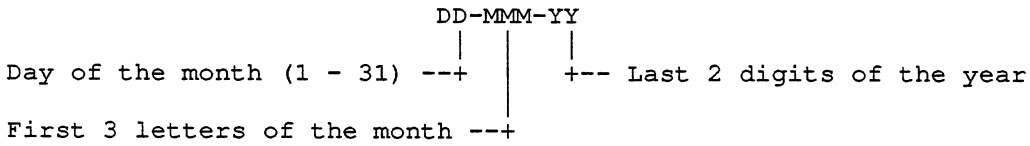
If communications cannot be established, the system supplies default settings for the items in the Scratch Pad. From the title screen, pressing any key displays the System Clock Setup screen. If the date and time shown are correct, press CTRL-E (or the Enter key) to display the Supervisor menu.

Changing the Date and Time

To change the date and time, type in entries using the keyboard or the numeric keypad. The date and time screen can only be accessed from the title display screen.

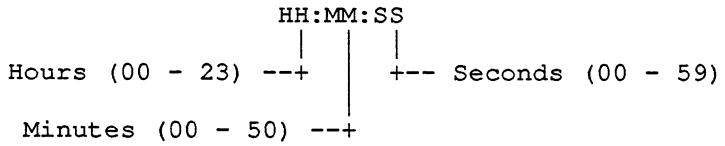


The date must be entered in the following format:



For example, a current date of April 2, 1990 would be entered as: 02-APR-90.

The time must be entered in the following format:

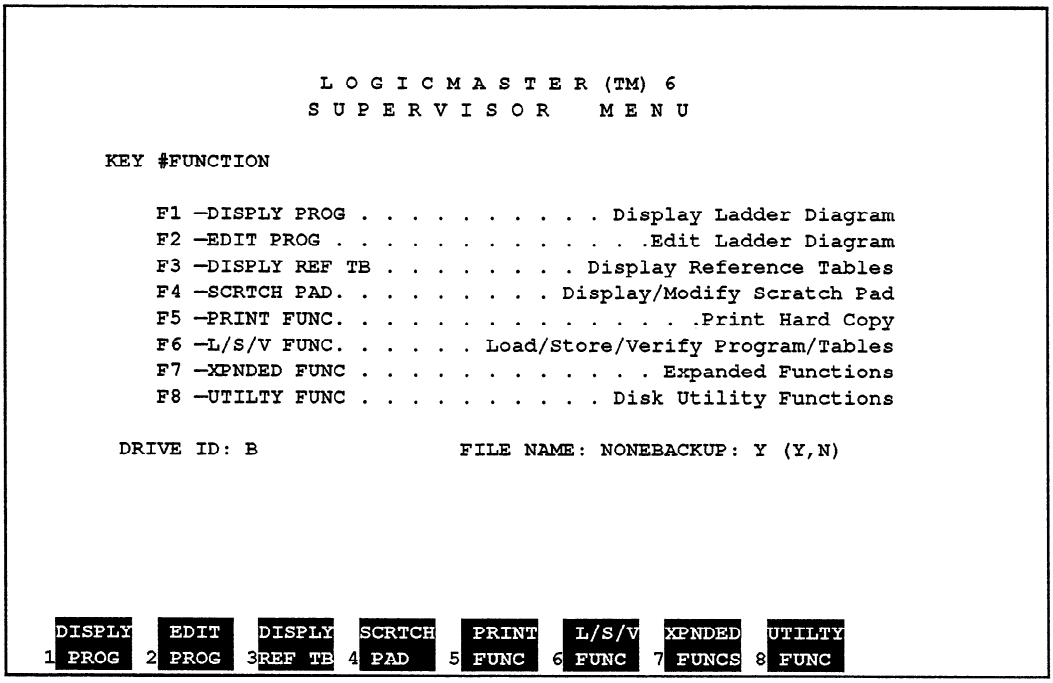


For example, a time of 2:30 p.m. would be entered as: 14:30:00.

Use the cursor keys (or Return key) to move between fields on the screen. When the time and date shown are correct, press CTRL-E (or the Enter key) to display the Supervisor menu. Pressing CTRL-E (or the Enter key) with no changes in the two fields accepts the current date and time as shown on the menu.

SECTION 4
Supervisor Menu

The Supervisor menu provides access to the primary functions of the Logicmaster 6 system. The menu lists the name and function of each function key.



The following function key assignments are available in the Supervisor menu.

Function Key	Function	Description
F1	Display Program	Display a ladder diagram program stored in system memory. If connected to an operating CPU, power flow and register content can be shown in On-Line or Monitor mode. Refer to chapter 4, *Display Program*, for information about this function.
F2	Edit Program	Develop a program or modify a program stored in the system. This function is performed within the Logicmaster system. It does not require connection to a CPU, and is available in Off-Line, On-Line, and Monitor mode. Refer to chapter 5, *Edit Program*, for information about this function.
F3	Display Reference Tables	Display I/O status or register contents. If connected to an operating CPU, real-time data can be displayed in On-Line or Monitor mode. In Off-Line mode, data is from the Logicmaster system internal memory status image. Refer to chapter 7, *Display Reference Tables*, for information about this function.
F4	Scratch Pad	Display information about memory size, function level, register capacity, CPU ID, and other status information. Use the Scratch Pad function to establish parameters before programming. Refer to chapter 3, *Scratch Pad*, for information about this function.
F5	Print Functions	Print out program and other information in either foreground or background mode. Refer to chapter 8, *Print Functions*, for information about this function.
F6	Load/Store/Verify Functions	Transfer programs between the Logicmaster 6 system, the CPU, and disks. Refer to chapter 9, *Load/Store/Verify*, for information about this function.
F7	Expanded Functions	Display or modify the CPU configuration data, or to clear GENIUS I/O faults. Refer to chapter 10, *Expanded Functions Menu*, for information about this function.
F8	Utility Functions	Copy diskettes and perform other file-handling operations. Utility functions are also used to clear parity errors when starting up a new CPU. Refer to chapter 11, *Utility Functions*, for information about this function.

The bottom of the Supervisor screen displays the following information:

Drive ID: This is the drive you are using for your program files. (Default = drive B.) To change it, enter the new drive ID followed by a file name or the word NONE. Press the Enter key on the numeric keypad (not the Return key). For example:

```
A:program1 (press the Enter key)
```

File Name: The program name.

Backup: Shows whether backup files (copies) will be made before program files are edited. File backup is selected by entering **Y** for the prompt that appears at the beginning of an editing session.

Loading Program Files from the Supervisor Menu

Program files can be loaded into Logicmaster 6 system memory using the Load/Store/Verify function, or directly from the Supervisor menu. If there is already a program in RAM, it must be cleared using the Load/Store/Verify functions before a new program can be loaded.

To load program files from the Supervisor menu:

1. Be sure that the file is present on the default drive (hard disk or diskette).
2. If the drive ID does not show the drive where the program files are located, change it. To enter a drive ID, type it into the work area and press the Enter key.
3. Enter the program file name. Type the basic program file name (for example, PROGRAM1), not including a file name extension into the work area. Then, press the Enter key.
4. Press the ALT and the L keys at the same time. All files with the file name entered will be loaded into RAM. The message "BUSY" appears on the screen until the files are loaded. If an error occurs, a message appears and the loading is aborted.

For information on using the Load/Store/Verify function to load files, refer to chapter 9, *Load/Store/Verify*.

Naming the Program

You should enter a program name before leaving the Supervisor level to edit, display, or print a program. The name is retained until a new program is loaded, or until the name is cancelled by entering the word "none", or until power is removed from the system.

If no name is entered, the system can develop programs, or monitor a Series Six CPU, but annotation cannot be used or displayed. In addition, without a program name, data values for registers, I/O, and overrides, and display formats will not be saved, because there will be no files to place them in. Finally, if no program name has been specified before editing a program, the system is unable to save the edits automatically. Edits will be saved in RAM. However, if power goes off during editing, the work you have done will be lost. Therefore, you should always enter the program name before leaving the Supervisor level.

A program name may have up to 8 characters. Do not use the following reserved words in program names: NONE, CON, NUL, PRN, AUX, COM1, COM2, LPT1, LPT2, LPT3.

SECTION 5
Starting up the Series Six PLC

This section explains how to set up serial communications between the Series Six programmable controller and the Logicmaster 6 system, and how to start up the Series Six PLC with the Logicmaster 6 software.

Setting up Serial Communications

With any serial version of Logicmaster 6 software, the computer communicates with the Series Six PLC over a serial link to a CCM card in the CPU rack. Communication can be at up to 19200 baud, with the CCM card set for zero millisecond turnaround delay.

The steps to set up communications in the Logicmaster 6 software are summarized below. For more information, refer to the appropriate sections of the manual. Hardware setup is described in appendix A, *Setup Information.*

Step 1: Programming the Serial Port (Utility Functions)

Initialize the computer's serial port (COM1 or COM2) for communications with the CCM card. This is done through the Serial Port Setup menu, part of the Utility functions. Select the port and baud rate. For CCM communications, you will normally use 8-bit data characters with ODD parity and one stop bit. The switches on the CCM card must be set accordingly.

You can save the selections you make as files on the Logicmaster 6 startup disk. If the selections are saved, they will be used automatically the next time you start up the Logicmaster 6 software. Since the Serial Port Setup utility may be used to set up more than one port, you must specify the name of the file into which the setup parameters are to be stored. For the COM1 port, use the file name PORT1.PSU. For the COM2 port, use the file name PORT2.PSU.

Step 2: Selecting the Protocol, Enabling On-Line Changes (Expanded Functions)

Select the communications protocol (master/slave or peer-to-peer) on the Communications Setup menu of the Expanded functions. Again, specify the serial port (COM1 or COM2) that will be used. Also select the desired protocol and the ID number of the Series Six CPU. The CCM card must have its switches set for the proper protocol. The CCM card must be reset after changing the switch settings.

If you will be changing data on-line to the Series Six CPU, you must enable on-line changes in this menu.

The Communications Setup menu saves its parameters in a file named COMSET.SET.

Step 3: Changing Operating Modes (Keyswitch/Scratch Pad Functions)

For the Workmaster and Cimstar I industrial computers, a three-position keyswitch is provided for selecting the operating mode.

An IBM PC, PC-XT or PC-AT computer, when running Logicmaster software, always starts up in Off-Line mode. For these computers, go to the Scratch Pad display after starting up the software if you want to change to On-Line or Monitor mode.

Starting up the Series Six PLC

Prior to startup, the Series Six PLC should be set up according to the installation instructions provided in this book and in the Series Six PLC manual. Continue at the heading below that reflects the type of communications between the Series Six PLC and the computer. This should be the same as the version of Logicmaster 6 software you are using (either serial or parallel).

Serial Communications

After setting up serial communications as described above, follow these steps using the serial version of the Logicmaster 6 software:

1. Place the Logicmaster 6 system in Off-Line mode.
2. Clear Logicmaster 6 system memory.
3. Verify that the system contains the correct function level of the CPU. From the Supervisor menu, select Scratch Pad (F4). Then, return to the Supervisor menu.
4. Turn the mode keyswitch on the Series Six PLC from RUN position to STOP position. Cycle power to the Series Six PLC.
5. Store a null program from the Logicmaster 6 system memory to the CPU.
   A. From the Supervisor menu, select Load/Store/Verify (F6).
   B. In the L/S/V menu, press F2 (Store). On the Store Program/Tables screen, enter **P** (for CPU) as the Drive ID and press CTRL-E (or the Enter key).
6. Clear parity errors in the Scratch Pad and Transition Table memories using the Clear Parity function. From the Supervisor menu, select Utilities (F8). From the Utilities menu, press F7 (Clear Parity) and press CTRL-E (or the Enter key). Then, return to the Supervisor menu.
7. Start the CPU. Turn the keyswitch on the CPU to RUN, then to STOP, or cycle power to the CPU.

Parallel Connection

After establishing parallel connection between the Series Six PLC and the Logicmaster 6 system, as described in appendix A, follow these steps using the parallel version of the Logicmaster 6 software:

1. Place the Logicmaster 6 system in Off-Line mode.
2. Clear Logicmaster 6 system memory.
3. Verify that the system contains the correct function level of the CPU. From the Supervisor menu, select Scratch Pad (F4). Then, return to the Supervisor menu.
4. Store a null program from the Logicmaster 6 system memory to the CPU.
   A. From the Supervisor menu, select Load/Store/Verify (F6).
   B. In the L/S/V menu, press F2 (Store). On the Store Program/Tables screen, enter **P** (for CPU) as the Drive ID and press CTRL-E (or the Enter key).
5. Clear parity errors in the Scratch Pad and Transition Table memories using the Clear Parity function. From the Supervisor menu, select Utilities (F8). From the Utilities menu, press F7 (Clear Parity) and press the Enter key. Then, return to the Supervisor menu.
6. Start the CPU. Turn the keyswitch on the CPU to RUN, then to STOP, or cycle power to the CPU.

SECTION 6
Using your Keyboard

This section explains:

- How to select a keyboard for use with the Logicmaster 6 programming and documentation software.
- How to use a personal computer keyboard for programming.

NOTE

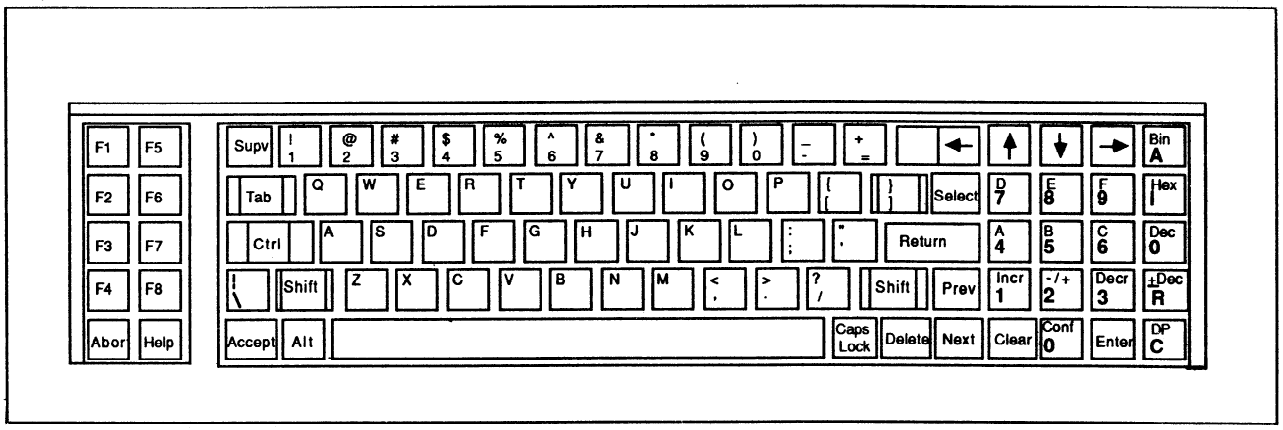
The 91-key keyboard described here was developed especially for use in ladder diagram programming. It has several keys that are not on standard personal computer keyboards. While a personal computer keyboard can be used for programming, as described in this book, the 91-key keyboard is recommended.

- ALT and CTRL key functions.
- How to define sequences of frequently-used keystrokes for easier programming.

91-Key Keyboard

The 91-key keyboard from GE Fanuc - NA was designed to satisfy the special requirements of PLC programming and monitoring. It has two modes of operation. In default mode, it operates as described on the following pages. It is compatible with Logicmaster programming software. Its extra keys perform special programming functions. In alternate mode, it emulates the functions of a standard 83-key IBM PC-XT keyboard.

a41680



Typewriter Keys

The central area of the keyboard includes the typewriter keys, in standard QWERTY layout. All of these keys will repeat their functions when held down.

The central area of the keyboard also includes these keys:

Key	Function
Suprv	The Supervisor key returns the display to the Supervisor menu.
Tab	The Tab key works like a standard typewriter Tab key. Tab stops are set every 8 characters.
Ctrl	The Control key, when pressed at the same time as another key, controls the execution of a function or command. The function performed depends on the software.
| / \	The Backslash key types a backslash character, or a split vertical bar when shifted.
Accept	The Accept key (or CTRL-A) is used to verify and accept an operation, such as a ladder diagram rung entry in the Edit Program function.
Alt	The Alternate key, when pressed at the same time as another key, controls the execution of a function or command. Refer to the heading "ALT Key Functions" in this section.
Caps Lock	The Caps Lock key locks the characters A through Z into uppercase. It does not lock the shifted characters of other keys.

Left Keypad

The keypad on the left includes keys labelled F1 through F8. The functions of these are controlled by the Logicmaster 6 software. The currently-active functions are displayed at the bottom of the screen. The left keypad also includes these keys:

Key	Function
Abort	The Abort key (or F9) is used to end the current operation.
Help	The Help key (or F10) is used to display the Help text associated with the current operation taking place in the software.

Right Keypad

The right keypad contains many of the keys you will use most frequently during programming with the Logicmaster 6 system.

Key	Function
Arrow keys	The arrow keys move the cursor.
Backspace	The Backspace key backs up the cursor and deletes any characters the cursor moves over.
Del	The Delete key (or CTRL-D) erases the previous/last character.
Next	The Next key (or CTRL-N) scrolls the program display upward to the next rung.
Enter	The Enter key (or CTRL-E) places the value currently in the work area into the program.
Prev	The Previous key (or CTRL-P) scrolls the program display downward to the previous rung.
Select	The Select key (or CTRL-S) moves the reverse video banner in the work area of the screen. This allows data to be entered in the text line, reference line, or value line of the work area. For more information, refer to the explanation on screen format.
Data Entry Keys	The gray keys in the right keypad have dual functions, as described in the next table. Unshifted, they are used to enter values into the work area during programming. Shifted, they change the number base of the work area, or of a selected item in one of the display tables.

Key	Shifted	Unshifted
Bin/A	Binary display	A (for AI or AO)
Hex/I	Hexadecimal	I (for Input)
Dec/O	Unsigned decimal	O (for Output)
± Dec/R	Signed decimal	R (for Register)
DP/C	Double-precision	C (for Constant)
Conf/0	Confirm action	Zero
Incr/1	Increment by one	1
-/+/2	Toggle sign	2
Decr/3	Decrement by one	3

Alternate Mode of the 91-Key Keyboard

In default mode, the keys of the 91-key keyboard execute their normal functions. The 91-key keyboard can also emulate the functions of an 83-key keyboard, as described below.

Pressing the CTRL and Select keys at the same time toggles the keyboard between 91-key and 83-key mode. Note that these functions are not used with the Logicmaster 6 software. This information is included here for reference only.

91-Key Mode	83-Key Mode
Accept	Alternate
Left Cursor	Backspace (erases)
Right Cursor	Scroll Lock
Up Cursor	Num Lock
Down Cursor	Num Lock
Decimal/0	Plus bar
Dec/R	Plus
Clear	0/Ins
Conf/0	0/Ins
Return	New Line
Enter	Delete/.
Incr/1	1/End
-/+/2	2/Down Cursor
Decrement/3	3/Page Down
A/4	4/Left Cursor
B/5	5
C/6	6/Right Cursor
D/7	7/Home
E/8	8/Up Cursor
F/9	9/Page Up
Suprv	Escape
Tab	Tab
Select	Tilde/Accent Grave
Abort	F9
Help	F10
DP/C	Plus
Prev	Print Screen/*
Next	Minus
Binary/A	Scroll Lock
Hexadecimal/I	Minus
Delete	Delete/.

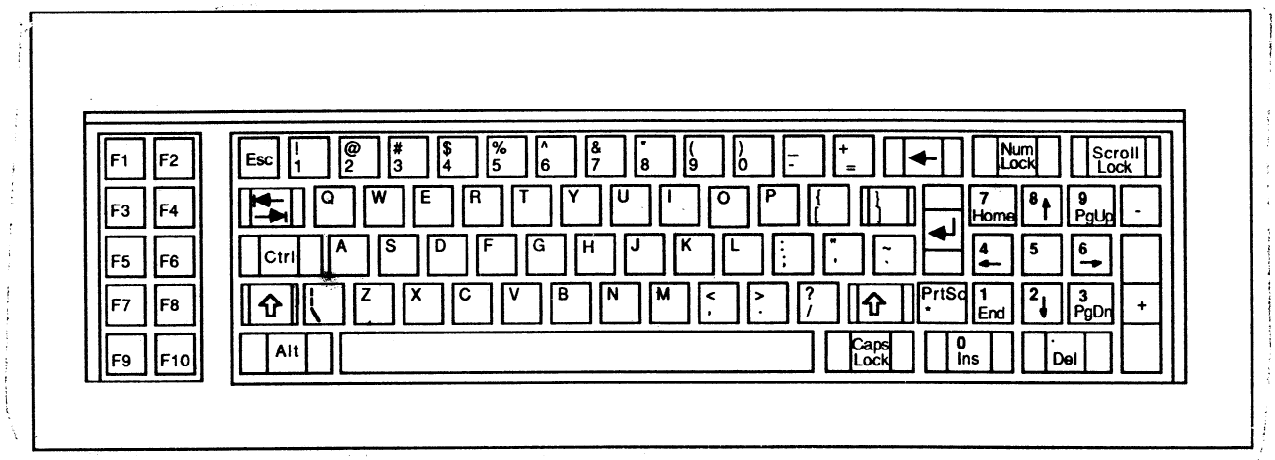
Personal Computer Keyboard

The following information describes the use of a standard IBM personal computer-type keyboard with Logicmaster 6 software. Note that such a keyboard lacks the additional programming keys of the 91-key keyboard. Therefore, the 91-key keyboard is recommended.

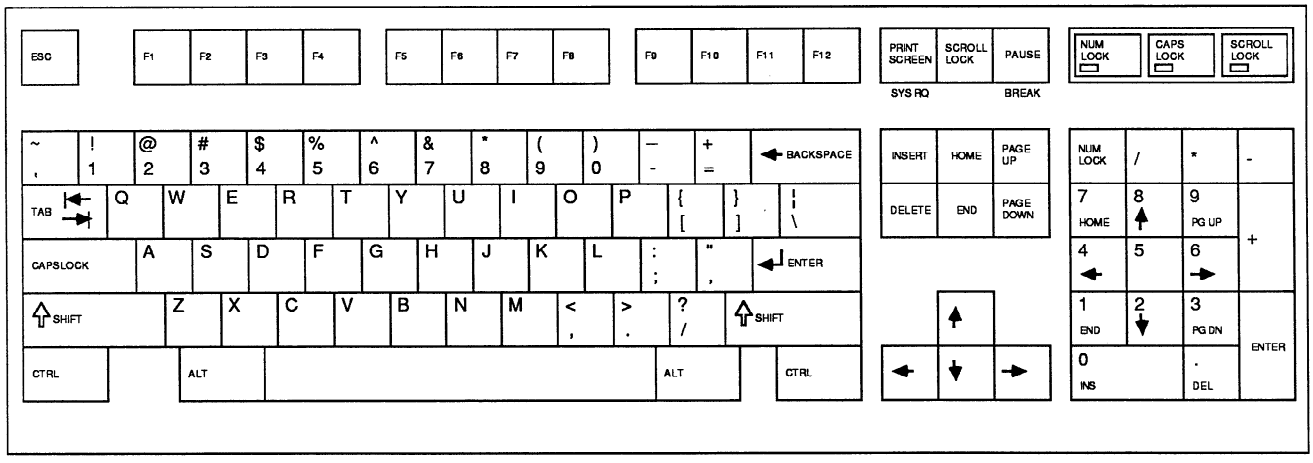
The first illustration below shows the layout of the IBM PC-XT keyboard. This is the same layout as the 83-key keyboard from GE Fanuc - NA.

The second illustration shows the layout of the Workmaster II computer keyboard. The Workmaster II and IBM PC-AT keyboards have a different layout, but are functionally similar.

a41677



a43110



Typewriter Keys

The central area of the keyboard includes the typewriter keys, in standard QWERTY layout. All of these keys will repeat their functions when held down.

The central area of the keyboard also includes these keys:

Key	Function
Esc	The Escape key emulates the function of the Supervisor key. It returns the display to the Supervisor menu.
Ctrl	The Control key, when pressed at the same time as another key, controls the execution of a function or command. The function performed depends on the software.
| / \	The Backslash key types a backslash character, or a split vertical bar when shifted.
Alt	The Alternate key, when pressed at the same time as another key, controls the execution of a function or command. Refer to the heading "ALT Key Functions" in this section.
*** / PRT SC**	The Print Screen key can be used to print out a copy of what is currently displayed on the screen. The printing parameters must previously have been established, as described in this manual. Print Screen is a shifted function.
Caps Lock	The Caps Lock key locks the characters A through Z into uppercase. It does not lock in the shifted functions of other keys.

Left Keypad

The keypad on the left includes keys labelled F1 through F10. The functions of these keys are controlled by the Logicmaster 6 software. Keys F1 through F8 function as shown by the labels that are displayed on the bottom of the screen.

Key	Function
F9	The F9 key functions as the Abort key. It is used to end the current operation.
F10	The F10 key functions as the Help key. It is used to display the Help text associated with the current operation taking place in the software.

Right Keypad

The right keypad contains many of the keys you will use most frequently during programming with the Logicmaster 6 system.

Key	Function
Num Lock	The Num Lock key locks the keys in the numeric keypad into their "upper" functions. For example, with the Num Lock key on, the cursor keys can be used to enter the numbers printed on the keys.
Arrow keys	With Num Lock off, the arrow keys move the cursor.
Backspace	When used with the Logicmaster 6 software, the Backspace key emulates the 0/Dec key on the 91-key keyboard. This key is used to enter a letter O (for output) in the center line of the work area, or to change the bottom line of the work area to decimal format.
Del	The Delete key (or CTRL-D) erases the last character entered.
Ins/0	With Num Lock off, the Ins/0 key functions as the Next key. It scrolls the program display upward to the next rung. With Num Lock on, this key enters a zero.
PgDn/3	With Num Lock off, this key functions as the Enter key. It places the value currently in the work area into the program. With Num Lock on, this key enters a number 3 in the work area.
Prt Sc / *	Unshifted, the Print Screen/* key functions as the Previous key. It scrolls the program display downward to the previous rung.
~ / '	Unshifted, the Tilde key functions as the Select key. It moves the reverse video banner in the work area of the screen. This allows data to be entered in the text line, reference line, or value line of the work area. For more information, refer to section 4, *Screen Format.*

For more information, refer to the following list of key functions.

Key Functions

The following table describes the key functions of the 91-key keyboard, in conjunction with Logicmaster 6 software. These same key functions are available on a personal computer keyboard through the use of Control key sequences, as listed in the third column of the table. Control key sequences and Alternate key sequences are produced by pressing the CTRL key or the ALT key at the same time as another key.

Table 2-1. Key Functions

91-Key Keyboard for the Workmaster Computer	Function Description	PC-Type Keyboard
Supervisor (Suprv) key	Returns the display to the Supervisor menu	**Escape (Esc) key**
Control (CTRL) key	When pressed at the same time as another key, controls the execution of a function or command. The function performed depends on the software.	**Control (CTRL) key**
Alternate (ALT) key	When pressed at the same time as another key, controls the execution of a function or command.	**Alternate (ALT) key**
Abort key	Ends the current function or operation.	**F9 function key**
Help key	Displays the Help screen associated with the current operation taking place in the software.	**F10 function key**
Accept key	Verifies and accepts an operation, such as entering the rung of a ladder diagram in the Edit Program function.	**CTRL-A**
Caps Lock key	Locks the characters A through Z into uppercase letters. However, it does not lock in the shifted functions of other keys.	**Caps Lock key**
Arrow keys	Move the position of the cursor.	**Arrow keys with Num Lock off**
Select key	Moves the reverse-video banner in the work area of the screen. This allows data to be entered in the text line, reference line, or value line of the work area. For more information, refer to the explanation of the screen format.	**CTRL-S**
Previous (Prev) key	Scrolls the program display downward to the previous rung.	**CTRL-P**
Next (Next) key	Scrolls the program display upward to the next rung.	**CTRL-N**
Delete key	Erases the previous/last character.	**CTRL-D**
Clear key	Clears the work area line.	**CTRL-Z**
Confirm (Conf) key	Confirms the prompt.	**ALT-X**
Enter key	Enters the current function. Places the value currently in the work area into the program.	**CTRL-E**

Table 2-1. Key Functions - Continued

91-Key Keyboard for the Workmaster Computer	Function Description	PC-Type Keyboard
(no equivalent key)	Locks the keys in the numeric keypad into their "upper" functions. For example, with the Num Lock key on, the cursor keys can be used to enter the numbers printed on the keys.	**Num Lock key**
	Data Entry Keys	
	Unshifted, the Data Entry keys are used to enter values into the work area during programming:	
Bin/A	Auxiliary or Indirect register references	**CTRL-U**
Hex/I	Input references	**CTRL-I**
Dec/O	Output references	**CTRL-O**
±Dec/R	Register references	**CTRL-R**
DP/C	Constants	**CTRL-C**
	Shifted, the Data Entry keys change the number base of the work area, or of a selected item in one of the display tables:	
Bin/A	Binary	**CTRL-B**
Hex/I	Hexadecimal	**CTRL-H**
Dec/O	Unsigned Decimal	**CTRL-,**
±Dec/R	Signed Decimal	**CTRL-/**
DP/C	Double Precision	**CTRL-.**

Alternate Key Functions

Alternate key functions provide a quick and simple method of performing various software functions with a minimum of keystrokes. These key functions are available on the 91-key keyboard, as well as the personal computer-type keyboard, by pressing the ALT key (or the CTRL key where noted) at the same time as another key. The following table provides a description of these ALT key functions.

Table 2-1. Alternate Key Functions

Key(s)	Function Description
	All Modes
ALT-P	Print screen. (The printer must already be set up, and background printing must be enabled. Refer to chapter 8, *Print Functions.*)
ALT-X	Confirm a prompt.
ALT-1	For personal computers. Emulates the keyswitch function. At startup, the computer is in Off-Line mode. Pressing ALT-1 toggles the mode to Monitor, then to Off-Line, and then back to On-Line mode.
ALT-2	Start or stop the CPU from anywhere in the software, if the CPU keyswitch is not in Stop mode.
	Supervisor Menu
ALT-L	Load file without entering L/S/V functions.
ALT-Z	Exit to DOS. (Press CTRL-ALT-DEL or type **LM6** to return.)
ALT-C/DP	Where floating point format is used, change the work area value line to floating point.
	Display Program Mode
ALT-E	Go to Display Comments function to display rung explanation.
ALT-O	Go to On-Line Change menu.
ALT-U	Turn on Fast Update (serial versions only).
	Display Reference Tables
ALT-A	Display register contents in ASCII format (Expanded functions).
ALT-C/DP	For register references, change the work area value line and screen display to floating point format.
	For discrete references, change only the work area value line to floating point. (Does not change the screen display format.)
ALT-U	Turn on Fast Update (serial versions only).

Table 2-2. Alternate Key Functions - Continued

Key(s)	Function Description
	Edit Program Mode
ALT-D	Delete multiple rungs.
ALT-G	Read the side file named in the work area.
ALT-W	Write the side file named in the work area.
	Edit Ladder Diagram Rung
ALT-A	Display Basic Arithmetic keys.
ALT-B	Display Bit Matrix keys.
ALT-C	Display Control Function keys.
ALT-D	Display Data Move keys.
ALT-H	Display Shift/Move keys.
ALT-I	Display Timer/Counter keys.
ALT-M	Display Matrix Function keys.
ALT-R	Display Relay keys.
ALT-S	Global Reference Relay.
ALT-T	Display Table Move keys.
ALT- —	Display Expanded Arithmetic keys.
	Search Mode
ALT-A	Display Basic Arithmetic keys.
ALT-B	Display Bit Matrix keys.
ALT-C	Display Control Function keys.
ALT-D	Display Data Move keys.
ALT-F	Display List Function keys.
ALT-H	Display Shift/Move keys.
ALT-I	Display Timer/Counter keys.
ALT-M	Display Matrix Function keys.
ALT-O	Display On-Line Changes keys.
ALT-R	Display Relay keys.
ALT-S	Display Special function keys.
ALT-T	Display Table Move keys.
ALT- —	Display Expanded Arithmetic keys.

Expanded I/O

To enter Expanded I/O channels in the work area reference line, press the CTRL key with a key from the numeric keypad which represents the channel number:

```
CTRL-0 through CTRL-9 and
CTRL-SHIFT 4 through CTRL-SHIFT 9 for channels 0 through F.
CTRL-CLEAR Clear the channel if one exists.
```

Customized Key Functions (Teach Mode)

The system can save sequences of keystrokes, and repeat all of the keystrokes at a press of the ALT key and a function key (F1 through F8). The saved keystrokes might represent a series of frequently-used functions, or part of the program that you want to duplicate.

CAUTION

Do not include keystrokes representing Pause, Resume, or Abort functions. Use of these keys may lead to unpredictable results when using customized key functions.

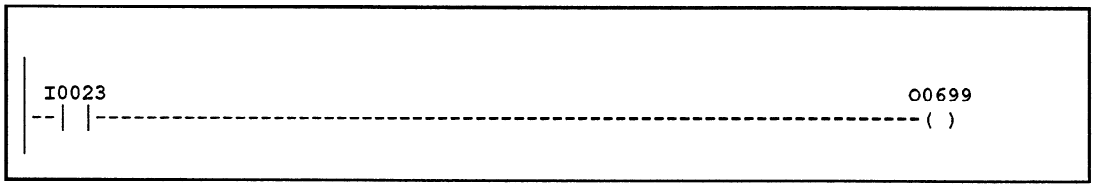
The keystrokes that make up the customized function will be stored in a file Fx.DEF, where x is the number of the function key. A diskette can store up to eight of these customized key functions - one for each key F1 through F8. Additional key functions can be stored on other diskettes.

Creating a Customized Key Function

To create a customized key function:

1. Place the system in the exact mode and condition for the function to begin. The system will remember this configuration, and not allow the function to be used later unless the configuration matches.
2. It is best to begin the function at a basic level, such as the Supervisor menu, Print menu, or Edit Program. When you are at the exact point for the function to begin, press ALT-J to place the system in Teach mode.
3. Press the ALT key and the function key (F1 - F8) that will control the function. The new function will replace the original function of the key in that location, so choose an unused key or one you will not need.
4. To create the function, you can use any succession of keystrokes. Do not include keystrokes for Pause, Resume, or Abort. The system records every keystroke until the ALT and K keys are pressed. After each 100 keystrokes, the system stores the key sequence in the program file.
5. While the system is in Teach mode, it records all keystrokes except those that can be used to end the function. To end the function normally and store it in the FX.DEF file, press ALT-K.
6. If an error condition is encountered during playback of the stored sequence, the sequence is halted. Pressing any key causes an error message to be displayed. Respond to the prompt that appears to either resume or terminate the key sequence.

As an example of a key function that might be defined, you could create the following simple line of ladder logic using Teach mode.



Using a Customized Key Function

After a key function is defined and stored in the FX.DEF file, it can be recalled by pressing the ALT key together with the function key assigned to that customized function.

For the function to be correct, the screen display and function key assignments must be the same as when the function was created. Cursor position, work area content, and data on the screen need not be the same. Be sure to check these before using the function.

When the customized function is selected, using the ALT key and the assigned function key, the system repeats the keystrokes very rapidly. During this "playback" of the function, the only keyboard input recognized is F9 (or the Abort key). Press F9 (or the Abort key) to stop the function; this will not, however, remove the part of the function that has already been performed. If an error condition is encountered during playback, an error message is displayed and the system aborts the function automatically.

Displaying/Printing a Customized Key Function (View Mode)

The system stores the defined function in the .DEF file as a list of the keystrokes that were executed for the function. This list can be displayed on the screen, and printed using the Print Screen command (ALT-P).

To display the function, press ALT-V. Then, press the ALT key together with the function key that is assigned to that function. The View Mode screen appears, listing the stored keystroke sequence.

For the line of ladder logic shown in the example above, the screen would display the following:

```
F6,004,009                V I E W   M O D E
F1,F1,INPUT,#2,#5,ENTER,F7,OUTPUT, #6,#9,#9,ENTER,F7,ENTER
```

For this function, the bottom of the View Mode screen shows the rung where the function was originally defined. The work area shows O 699.

A single View Mode screen can list approximately 500 to 600 keystrokes, separated by commas. If the entire function list does not fit on one screen in View mode, press the Next key to view additional screens.

In the previous example:

- The first key listed is the function key which has been redefined. In this example, it is F6.
- The next two numbers represent the display level where the function was defined. The displays represented by these numbers are listed on the following pages.
- On the next line, the listing of keystrokes begins.
  - Capital letters indicate that the shifted and unshifted values of the key are essentially equal. For example, A.
  - The up karet (^) character indicates a press of the CTRL key.
  - An exclamation point (!) indicates a press of the ALT key.
  - The number symbol (#) appears before numbers input from the numeral keypad. No number symbol appears before numbers input from the ASCII keyboard.
  - Keys such as ENTER or OUTPUT are abbreviated in the following table.

Keyname Abbreviations in View Mode	
Abbreviation	**Description**
F1 - F8	Function key F1 - F8
ABORT	Abort key
ACCEPT AUX	Auxiliary Reference key
CLEAR	Clear key
CONF	Confirm Upper Case Zero
CONST	Constant Reference key
DELETE	Delete key
DOWN	Cursor Down key
ENTER	Enter key
HELP	Help key
INPUT	Input Reference key
LEFT	Cursor Left key
NEXT	Next key
OUTPUT	Output Reference key
PREV	Previous key
REG	Register Reference key
RETURN	Return key
RIGHT	Cursor Right key
SELECT	Select key
SUPRV	Supervisor key
UP	Cursor Up key

To end the View Mode display, press ALT-V, F9 (or the Abort key), or the Escape key (or the Supervisor key).

Number Values Representing Defined Function Starting Screens

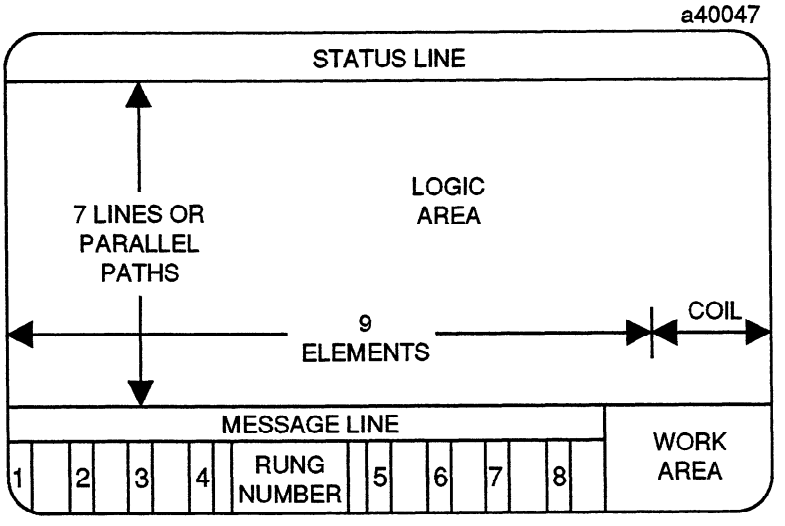
The following number pairs appear in the upper left of the View Mode screen, directly after the function key designation. These values represent the starting screen and key functions that were active when the function was defined. To use the defined function, the same screen and key functions must be active.

Values	Active Mode	Active Function Key Assignments
001,000	Edit Program	Page Mode Name Entry
003,002	Display Program	Display Program Menu
003,003	Display Program	Window Mode
003,004	Display Program	Page Mode
003,005	Display Program	On-line Changes
003,029	Display Program	Search Functions
003,030	Display Program	Search - Relays
003,031	Display Program	Search - Timers/Counters
003,032	Display Program	Search - Basic Functions
003,033	Display Program	Search - Shift/Move
003,034	Display Program	Search - Basic Arithmetic
003,035	Display Program	Search - Special Functions
003,036	Display Program	Search - Advanced Functions
003,037	Display Program	Search - Data Move
003,039	Display Program	Search - Expanded Arithmetic
003,040	Display Program	Search - DP Arithmetic
003,041	Display Program	Search - FP Arithmetic
003,042	Display Program	Search - Table Move
003,043	Display Program	Search - List Functions
003,044	Display Program	Search - Matrix Functions
003,045	Display Program	Search - Bit Matrix
003,046	Display Program	Search - Control Functions
003,047	Display Program	Search - Configuration Functions
004,006	Edit Program	Edit Program Menu
004,007	Edit Program	Window Mode
004,008	Edit Program	Page Mode
004,009	Edit Program	Basic Functions Menu
004,010	Edit Program	Relay Elements
004,011	Edit Program	Timers/Counters
004,012	Edit Program	Move/Shift
004,013	Edit Program	Basic Arithmetic
004,014	Edit Program	Special Functions
004,015	Edit Program	Advanced Functions
004,016	Edit Program	Data Move
004,018	Edit Program	Expanded Arithmetic
004,019	Edit Program	DP Arithmetic
004,020	Edit Program	FP Arithmetic
004,021	Edit Program	Table Move
004,022	Edit Program	List Functions
004,023	Edit Program	Matrix Functions
004,024	Edit Program	Bit Matrix
004,025	Edit Program	Control Functions
004,026	Edit Program	Configuration Functions
004,029	Edit Program	Search Functions
004,030	Edit Program	Search - Relays
004,031	Edit Program	Search - Timers/Counters
004,032	Edit Program	Search - Basic Functions
004,033	Edit Program	Search - Shift/Move
004,034	Edit Program	Search - Basic Arithmetic
004,035	Edit Program	Search - Special Functions
004,036	Edit Program	Search - Advanced Functions
004,037	Edit Program	Search - Data Move

Values	Active Mode	Active Function Key Assignments
004,039	Edit Program	Search - Expanded Arithmetic
004,040	Edit Program	Search - DP Arithmetic
004,041	Edit Program	Search - FP Arithmetic
004,042	Edit Program	Search - Table Move
004,043	Edit Program	Search - List Functions
004,044	Edit Program	Search - Matrix Functions
004,045	Edit Program	Search - Bit Matrix
004,046	Edit Program	Search - Control Functions
004,047	Edit Program	Search - Configuration Functions
009,054	Load/Store/Verify	Clear Memory
010,049	Load/Store/Verify	L/S/V/Menu
011,050	Load/Store/Verify	Load
012,051	Load/Store/Verify	Store
013,052	Load/Store/Verify	Verify
014,058	Print	Print Menu
015,059	Print	Print Out
016,060	Print	Print Program
017,061	Print	Define Printer
018,062	Print	Define Output
019,064	Display Tables	Input Table Display
020,064	Display Tables	Output Table Display
021,064	Display Tables	Auxiliary Input Table Display
022,064	Display Tables	Auxiliary Output Table Display
023,064	Display Tables	Expanded Input Table Display
024,064	Display Tables	Expanded Output Table Display
025,063	Display Tables	Register Data Display
026,063	Display Tables	Register Text Display
027,065	Display Tables	Mixed Table Display
028,066	Scratch Pad Display	On-line/Off-line
029,066	Scratch Pad Display	Scratch Pad Menu
030,067	Supervisor	Main Menu
031,068	Clock Set-Up	Time and Date Entry
032,069	Utility	Clear Parity Errors
033,070	Utility	Duplicate Master
034,071	Utility	Copy Files
035,072	Utility	Delete Files
036,073	Utility	Directory Listing
037,075	Utility	Port Setup
038,076	Utility	Utility Menu
039,077	Expanded Functions	CPU Configuration
040,078	Expanded Functions	Bus Controller Screen
041,079	Expanded Functions	Communications Setup
042,080	Expanded Functions	I/O Faults
043,081	Expanded Functions	Menu - parallel
044,082	Expanded Functions	Menu - serial
045,081	Expanded Functions	Series 90-70 I/O - parallel
046,082	Expanded Functions	Series 90-70 I/O - serial
047,083	Expanded Functions	Machine Setup Menu
048,084	Expanded Functions	Series 90-70 I/O Rack Configure
049,085	Expanded Functions	Series 90-70 I/O Rack Display

SECTION 7
Entering Data

The screen uses the following basic format:



The top line of the screen displays system status information. The central portion of the screen displays the program, menus, or data displays.

The bottom of the screen shows messages, and the current assignments of the function keys. Function key assignments vary depending on the function being used, and sometimes with the position of the cursor on the screen. These assignments are accessed with the keys labeled F1 through F8.

Status Line

The status line for the parallel version of Logicmaster 6 has the following format:

```
CPU: RUN/ENABLE  SWEEP: 123ms   W/M EQUAL CPU W/M: MONITOR  CURSOR: BA98
```

The Status Line for the serial version has the following format:

```
CPU: RUN/ENABLE     CPU ID: 2   W/M EQUAL CPU W/M: MONITOR  CURSOR: BA98
```

The Status Line for the IBM personal computer version has the following format:

```
CPU: RUN/ENABLE    CPU ID: 2  W/M EQUAL CPU NUM W/M: MONITOR  CURSOR: BA98
```

Definitions for the items on the status line include:

CPU Status: The first item shows the current status of the CPU. It may be RUN/ENABLE, STOPPED, or RUN/DISABLE.

CPU Sweep: For the parallel version, the second item shows the CPU sweep time in milliseconds. Sweep time depends on the type of Arithmetic Control Unit Module present in the CPU. When the CPU uses the Expanded II instruction set, the Arithmetic Control Unit Module (IC600CB524) must be present. This module sets the sweep time at 300 ± 50 ms. Previous versions of the module (used with the Basic, Extended, Advanced, and Expanded function sets) set the sweep time at 200 ± 50 ms.

CPU ID: For the serial versions, the second item shows the ID number of the CPU. Range = 1 to 90 for master/slave or 1 to 254 for peer-to-peer communications.

W/M-CPU Program: The third item shows whether the program in Logicmaster 6 system memory and in the CPU are EQUAL or NOT EQual.

NUM: This optional field appears when the Num Lock key is active (IBM PC version only).

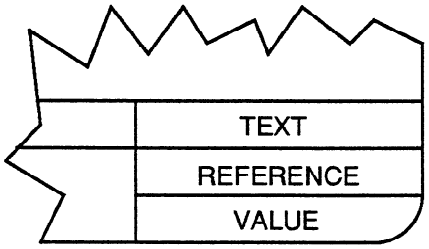
Mode: This item shows the current mode of the system with respect to the CPU. It may be: ON-LINE, OFF-LINE, or MONITOR. For the IBM PC version, go to the Scratch Pad display to set the mode.

Cursor Address: The last item shows the cursor address in the ladder diagram.

Work Area

The bottom right corner of the screen is referred to as the work area. It shows data that is typed in.

a40046



The top line of the work area is for text, such as program names. The top line is called the "text" line.

The second line of the work area contains reference numbers, such as I1024. This line is called the "reference" line.

The bottom line of the work area is used for values. The base of the entry (hex, decimal, signed decimal, and so on) can be changed by pressing the Reference Type keys. The bottom line is called the "data" line.

Only one line of the work area is active at a time, as shown by the reverse video block. To move between lines, press CTRL-S (or the Select key). On a personal computer keyboard, use the ('/tilde) key.

Entering Program References and Values

The following paragraphs explain how to enter program references and how to enter values for those references. This is how a typical program function, a DPREQ, looks on the screen when first entered in the logic.

```
  R****
-|DPREQ|-
```

The reference for this function must be a register, so the reference line begins with the letter R. You enter a reference for a function using the numeric keypad and press CTRL-E (or the Enter key). For example, suppose that the reference for this DPREQ is R0001. After entering the reference, the display looks like this:

```
  R0001
-|DPREQ|-
```

When CTRL-E (or the Enter key) is pressed, the cursor moves to the next position in the rung, where you could enter another logic function. For the DPREQ, however, you may also want to enter a "value" for the reference.

There are two ways to do that. The first way is to move the cursor back (using the Cursor Left key) to the function. Press CTRL-S (or the Select key) to move the work area cursor to the bottom line, and

enter the value for the reference. Here, suppose we enter the value 1993. After entering the value, press CTRL-E (or the Enter key). The value appears below the function on the display.

```
  R0001
-|DPREQ|-
  01993
```

The second way to enter a value for a program reference is easier, but should be used carefully. You can enter the reference and the value at the same time, and then press the Shift and Enter keys. Using the same example, here is the program function as it first appears:

```
  R0001
-|DPREQ|-
```

With the work area cursor at the center line, enter the reference for the function. Do not press CTRL-E (or the Enter key) yet. With the work area cursor at the bottom line, enter the value to be placed in the reference. When the entries are correct, press Shift-Enter to place both the reference and its value into the program.

```
  R0001
-|DPREQ|-
  01993
```

CAUTION

Do not get into the habit of using Shift-Enter for all programming. You may inadvertently enter incorrect values into registers which are also used to store I/O status.

Entering Conventional References

Enter references in the center line of the work area. To enter a reference other than an Expanded I/O reference:

1. If needed, use the Binary/A (Scroll Lock) key to toggle the first entry in the reference line between A and blank. For an auxiliary I/O reference, enter an A.
2. Enter one of the following:
   A. I for an Input or Auxiliary Input (Minus key).
   B. O for an Output or Auxiliary Output (Backspace key).
   C. R for a register (PgUp/9 key, Num Lock off).
   D. C for a constant (+ bar, Num Lock off).

NOTE

If the reference line was previously used for an Extended I/O reference, it may be necessary to press ALT-Clear first.

3. Use the numeric keypad to type in the reference. (For a PC keyboard, use the numeric keypad with Num Lock on).
4. After entering the reference, press CTRL-E (or the Enter key). For a register reference be careful not to press Shift and Enter, which enters any value in the data (bottom) line of the work area into the register. This may cause unexpected results in the program, especially if the register corresponds to Auxiliary Outputs.

Entering Expanded I/O References

To enter an Expanded I/O reference in the center line of the work area:

1. Enter I or O, as described above.
2. Press and hold the CTRL key. At the same time, enter the channel number (from 0 to F). Use the numeric keypad keys 0 to 9 for channels 0 to 9, and (shifted) 4 to 9 for channels A to F.

NOTE

Pressing CTRL-Clear removes the channel number field and the ± field from the reference line of the work area. If you are using a PC keyboard, refer to the Keyboard Function Summary in the previous section.

3. Press Shift and the ± 2 (Down Cursor, Numlock On) key to toggle the sign field. Enter a + or - sign, depending upon whether the I/O is real (+) or internal (-).
4. Enter the point address, which is a number from 00001 to 01024.

Entering Floating Point Values

To enter floating point numbers:

1. Set the work area numeric line to floating point format by pressing ALT-C. In floating point format, the numeric line is divided into two parts, the left part for the mantissa and the right part for the exponent of the floating point number. The cursor begins at the mantissa field:

   [+2.81660] -42

2. Enter characters as required, using the numeric keypad. As characters are typed, they enter the field from the right.
3. Press CTRL-S (or the Select key) to move cursor between the mantissa field and the exponent field:

   +2.81660 [-42]

   Enter the exponent. If none is entered, the system assumes a default exponent of +00.

NOTE

If the work area data line is in floating point format and the Clear key is pressed, the format defaults to decimal.

The Increment and Decrement keys do not function on a floating point value in the work area.

Entering a Floating Point Constant

To enter a floating point constant:

- Enter a C (+ bar, Num Lock off) on the reference line.
- Press CTRL-S (or the Select key) to move the work area cursor to the numeric line.
- If the numeric line is not already in floating point format, as shown above, press ALT-C.
- Enter the value as explained above.

Converting a Number to Floating Point Format

Any number in any format can be converted to floating point format. Double precision numbers lose some precision by this conversion, because only 7 or 8 significant digits are stored in floating point format.

Floating point numbers may not be converted to another base.

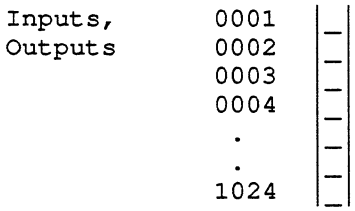
SECTION 8
Working with Numbers

Internally, a computer stores all numbers in binary. However, these numbers can be used in many ways, depending on the TYPE of data that is needed. To use the Logicmaster 6 system, you will need to understand the different number types used by the Series Six CPU, and how these numbers are handled.

Binary Data

In binary, data can be either a 0 or a 1. These two choices conveniently represent on/off conditions. All of the I/O data used in the system is in binary.

A binary digit is referred to as a BIT. It represents the smallest unit of data storage within memory. Inputs and outputs are stored in memory in adjacent bits. Each bit is numbered, beginning at 0001.

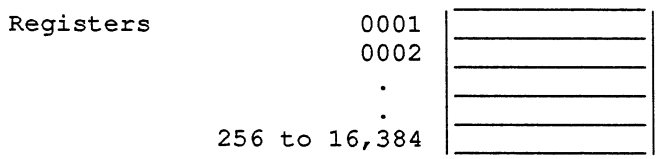


Bytes

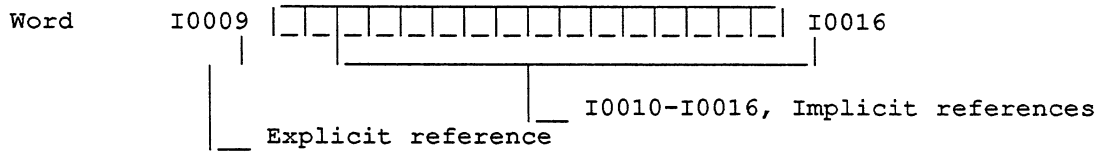
A group of 8 consecutive bits is referred to as a BYTE. A byte boundary marks the beginning of a new byte. The first byte boundary is reference 1, the second is 9, and so on to the last at 1017.

Registers and Words

A group of 16 consecutive bits may be referred to as either a register or a word. "Register" refers to a group of 16 consecutive bits located in register memory. The structure of these registers is fixed. Each register is numbered, beginning at 0001. The number of registers available depends on the amount of register memory available in the CPU.

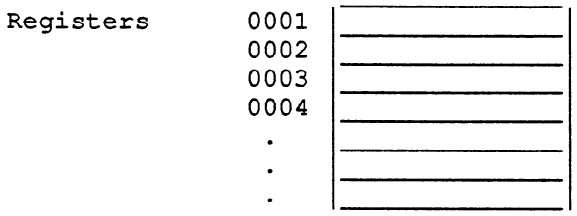


"Word" refers to a group of 16 consecutive bits in the Input or Output tables. The assignment of bits to words is done by several program functions. Valid word references occur on byte boundaries. Each word is referenced by the address of the most significant bit. In the illustration below, the explicit reference for the word is input I0009.



Decimal

Decimal numbers use 10 digits, 0 through 9. The system stores decimal numbers in binary format, in 16-bit registers or words.

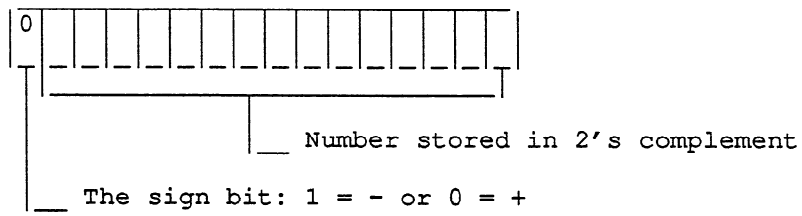


The range of decimal numbers that can be stored (in binary format) in one register depends on whether the number is signed, and whether it is a variable or a constant.

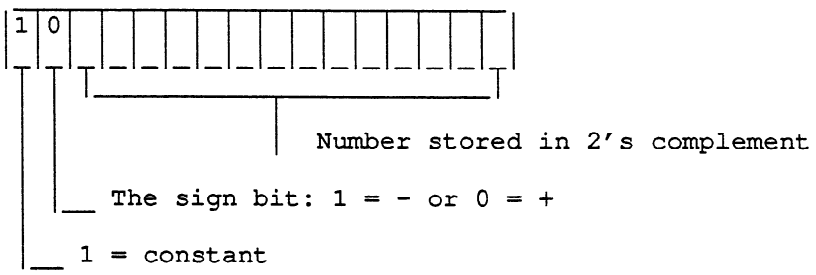
			Maximum	Minimum
Unsigned Binary	Single Precision	Variable Constant	65,535 32,767	0 0
Signed (2's Complement)	Single Precision	Variable Constant	+32,767 +16,383	-32,768 -16,384

The differing values occur because of the way the number is stored within the 16 bits of the register or word.

If the number is a variable, the leftmost bit of the implied reference is the sign bit.

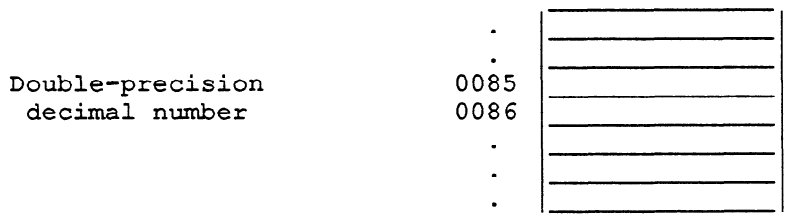


If the number is a constant, the leftmost bit is set to 1, leaving one less bit available to store the number. If the number is signed, the leftmost available bit becomes the sign bit. Therefore, the largest constant value that can be stored in single precision format is -16,383 to +16,384.



Double Precision Decimal

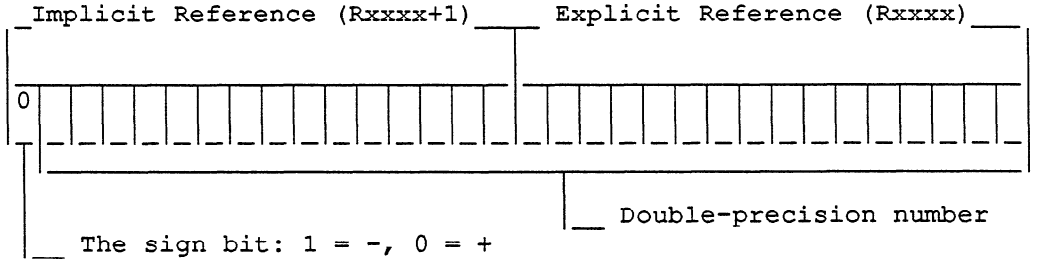
If a larger decimal number is needed, the system can store the value in two adjacent registers. That provides 32 bits for storing the number in binary format, as illustrated below. The assignment of adjacent registers to double precision storage format is made by the program. In the example shown, the first register selected for double precision format is 0085.



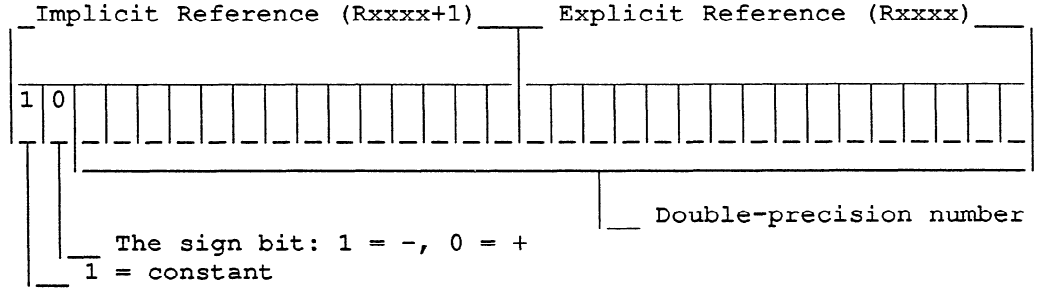
The first (lowest numbered) register is called an EXPLICIT reference, because it is explicitly identified. The second register is called an IMPLICIT reference, because its address is implied, although not explicitly given. The range of decimal numbers that can be stored (in binary format) in two registers depends on whether the reference is a variable or a constant.

			Maximum	Minimum
Signed	Double Precision	Variable	+2,147,483,647	-2,147,483,648
		Constant	+1,073,741,823	-1,073,741,824

If the number is a variable, the leftmost bit of the implied reference is the sign bit.



If the number is a constant, the leftmost bit is set to 1, and the next most significant bit is used as the sign bit.



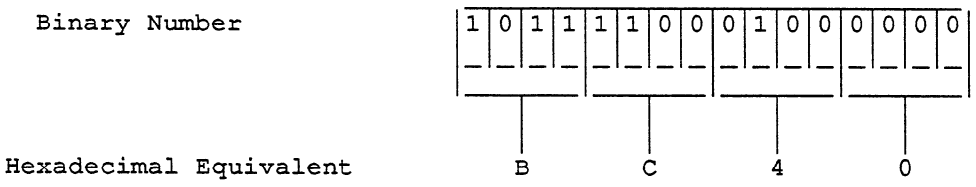
Hexadecimal

Hexadecimal uses 16 digits, 0 through F. The digits 0 through 9 represent their exact binary equivalent. The digits 10 through 16 are represented by the letters A through F.

Hexadecimal allows eight binary digits to be represented by two hexadecimal symbols. The table below shows the binary equivalents of the 16 hexadecimal digits.

Hexadecimal	Binary			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

Therefore, the binary value contained in a 16-bit register or word can be represented (and displayed or printed) as 4 hexadecimal digits. For example:



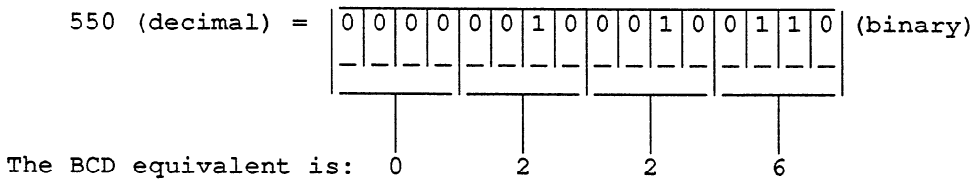
Binary Coded Decimal

Binary Coded Decimal (BCD) represents the decimal digits 0 through 9 as their binary equivalents. In BCD, one decimal digit is represented by 4 bits. Because this equivalence is on a single-decimal digit basis, the resulting number is not a true binary number (unless it is less than 9). Equivalent decimal and binary values are shown below.

Decimal	Binary			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
(none)	1	0	1	0
(none)	1	0	1	1
(none)	1	1	0	0
(none)	1	1	0	1
(none)	1	1	1	0
(none)	1	1	1	1

Because the decimal numbers from 10 to 15 would require two decimal digits, the last six bit patterns shown above are not available in BCD.

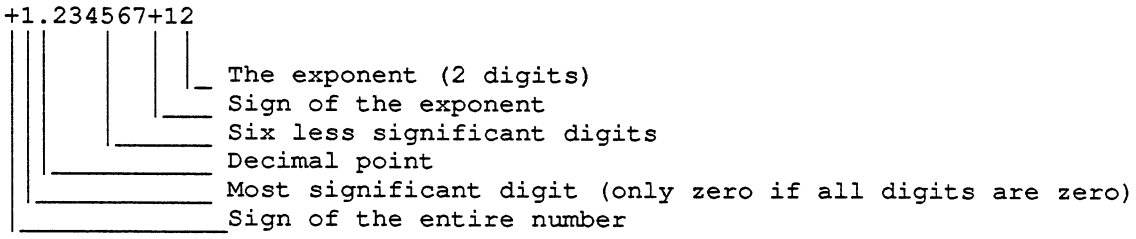
As an example, the number 226 is represented in BCD format as shown below. This is binary value is actually equal to the decimal number 550.



Floating Point

With the Expanded functions, floating point arithmetic can be used to perform the functions of addition, subtraction, multiplication, division, and compare. Floating point arithmetic allows efficient processing of fractional numbers, and a larger range of values than are possible with double precision numbers.

Floating point numbers are represented in decimal scientific notation, using seven significant digits. In the Series Six PLC, this has the following format:



In this format, the decimal point is placed after the most significant digit of the number. Leading zeros are not used. The exponent is a signed power of 10. Its value is equal to the number of places the decimal point must be moved to locate it after the first significant digit.

For example, 1234567 is equal to 1.234567×10^6. In floating point format, this can be represented as 1.234567+06.

If the conversion moves the decimal point to the left, the exponent has a positive sign. If the conversion moves the decimal point to the right, the exponent has a negative sign. Thus, the decimal number .0001234567 in floating point format becomes 1.234567-04.

More examples include:

```
+1.234567 = +1.234567+00      -12345.67 = -1.234567+04
-12.34567 = -1.234567+01      +.1234567 = +1.234567-01
+1234.567 = +1.234567+03      -.0000023 = -2.300000-06
```

All floating point operations are performed as if the numbers were correct to infinite precision. The result of the operation is rounded to the nearest number, or to even in case of a tie. Therefore, the rounded result will differ from the infinite exact result by at most 1/2 in the least significant position.

Values of Floating Point Numbers

The table below shows the possible values of floating point numbers. In the table:

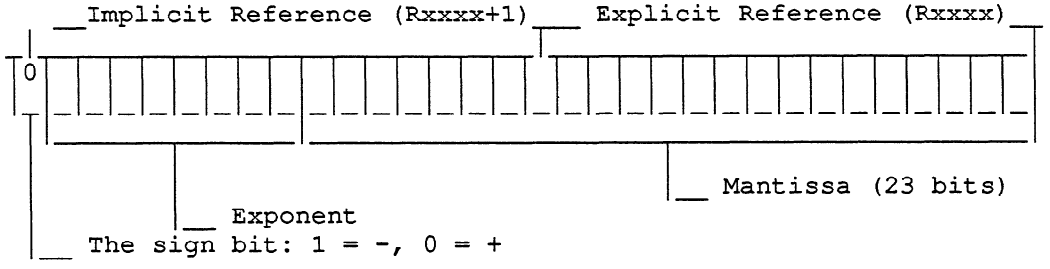
f	Represents the base 2 mantissa.
e	Represents the power of 2 exponent.
s	Represents the sign bit.

Exponent (e)	Mantissa (f)	Value of the Number
255	< > 0	NaN (Not a Number)
255	0	(-1)**s * Infinity
0 < e < 255		(-1)**s * 2**(e-127) * (1.f)
0	< > 0	(-1)**s * 2**(-126) * (0.f)
0	0	(-1)**s * 0, (zero)

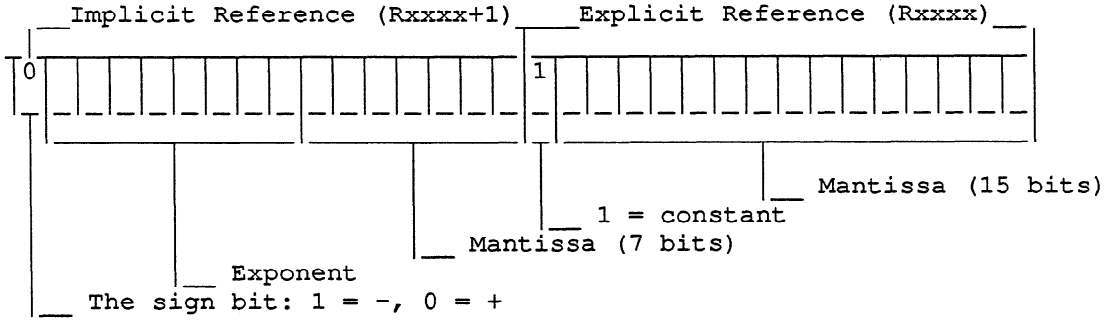
Storage of Floating Point Numbers

A floating point number is stored in IEEE 32-bit single-precision format in two adjacent registers of memory. The first register or bit address is the explicit reference. If an operand to a floating point operation is not a number, it is placed in a register but is not in IEEE format.

If the number is not a constant, the leftmost bit of the implicit reference is the sign bit. The range of reference-based floating point numbers is $\pm(1.401298 \times 10^{-45}$ to $3.402823 \times 10^{+38})$.



If the number is a constant, the leftmost bit of the explicit reference is set to 1. This reduces the precision of the fraction by one unit. The range of constant floating point numbers is $\pm(2.802596 \times 10^{-45}$ to $3.402823 \times 10^{+38})$.



Example

The register pair R15,16 contains the following bit pattern:

R16	R15
0100 0000 1010 0010	0000 1000 0000 0000

The Logicmaster 6 software displays this as:

```
40A2h           0800h           (in hexadecimal)
16546           2048            (in decimal)
+5.063476 +00                   (in floating point)
```

To calculate the floating point value, first determine: s, e, and f.

s = 0
e = 100 0000 1
f = 0100 0100 0001 0000 0000 0000

$n = (-1)^0 (2)^{129-127} (1 + 2^{-2} + 2^{-6} + 2^{-12})$
$n = (1) (2^2) (1 + .25 + .015625 + .000244140625)$
$n = (1) (4) (1.265869140625)$
$n = 5.0634765625$

Chapter 3
Scratch Pad

GEK-25379

The Scratch Pad function is used to store and display information about the current program in system memory, or about the CPU capabilities. Select the Scratch Pad function in order to:

- Change the ID number of the CPU.
- Display the current CPU capabilities in On-Line or Monitor mode.
- Compare the current CPU parameters with those stored for a program.
- Set up the current mode of the system (Off-Line, On-Line, or Monitor) for an IBM personal computer.
- Select the functions to be used when programming in Off-Line mode.

NOTE

When you are developing a program in Off-Line mode, establish the parameters in Scratch Pad before beginning. This will determine which functions are available during programming.

Setting System Mode with the IBM PC Version

Because IBM personal computers lack the mode keyswitch found on the Workmaster and Cimstar I computers, mode must be set through the software. This can be done in either of two ways:

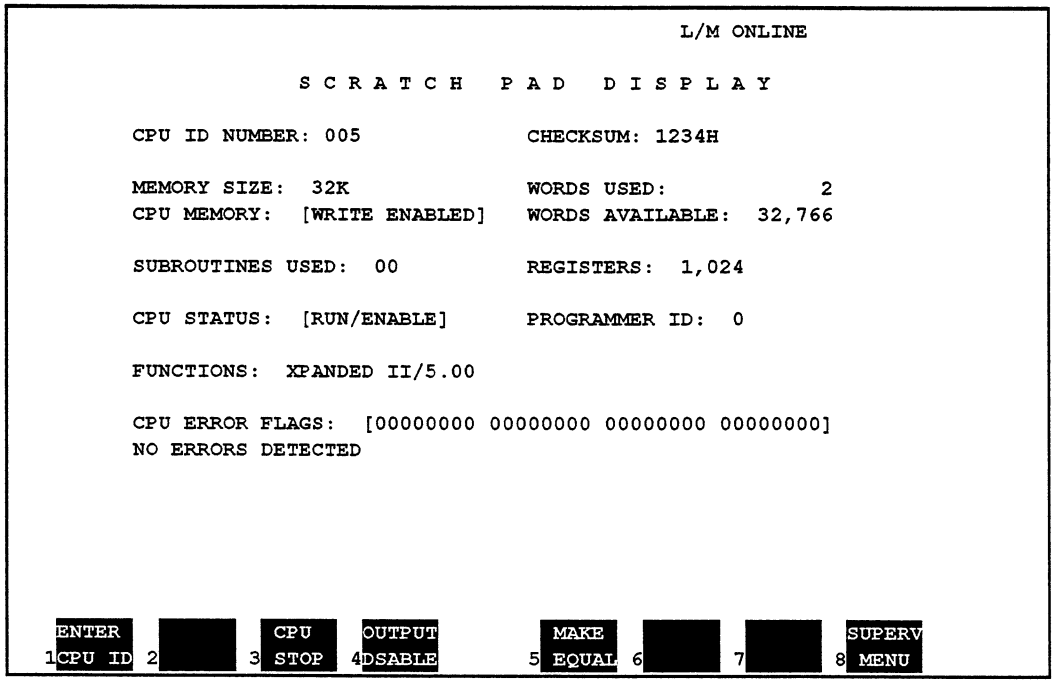
1. By pressing the ALT and 1 keys simultaneously. Repeatedly pressing ALT-1 switches the operating mode from Off-Line to Monitor to On-Line, and then back to Off-Line.
2. By entering the desired mode on the Scratch Pad display. To do this:
   A. On the Scratch Pad Display, type (into the screen's work area) the abbreviation of the mode you want to establish:

ON	=	On-Line mode
OFF	=	Off-Line mode
MON	=	Monitor mode

   B. Press the L/M Mode (F7) function key.

Accessing the Scratch Pad Display

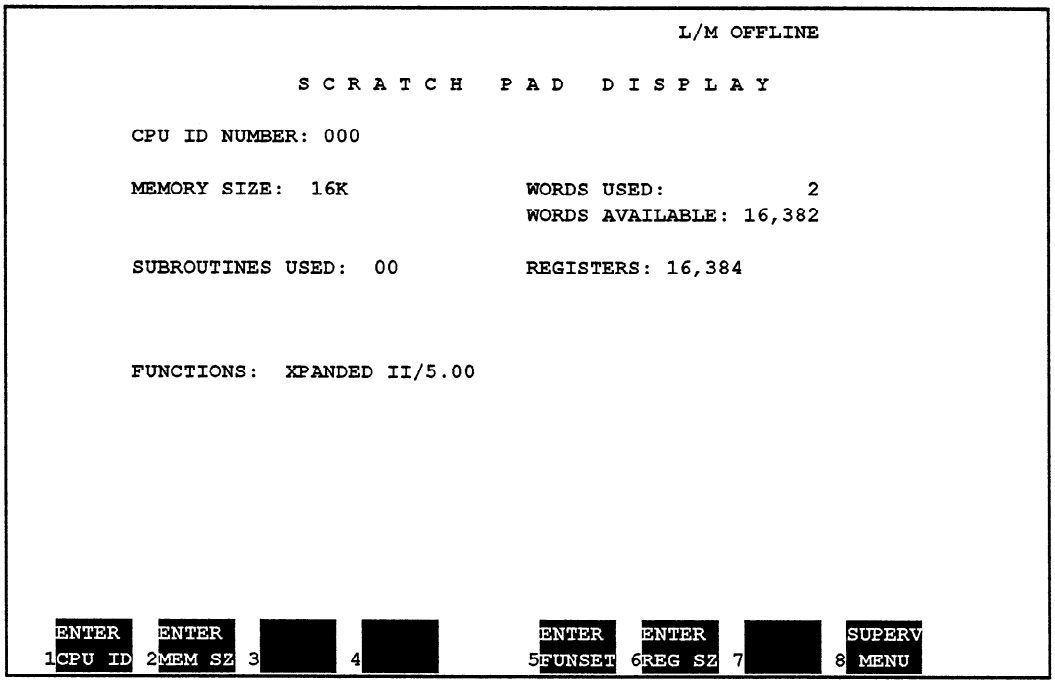
To access the Scratch Pad Display, select Scratch Pad (F4) from the Supervisor menu.



This screen shows the Scratch Pad display as it appears in On-Line or Monitor mode. In Monitor mode, function keys F1, F3, and F4 are inactive.

If a CPU is connected (scanning or not scanning) and the system is in On-Line or Monitor mode, the initial information on this screen comes from the CPU. Otherwise, it comes from the programming system's memory and the CPU-related entries are blank. In On-Line mode, when a connected CPU has capabilities that are different from the settings in the system, its differing values are shown in reverse video beside the values stored by the system.

In Off-Line mode, information about the CPU (version number, memory protect, error flags, and Run/Stop switch) is not displayed on the screen. The user program checksum and programmer ID are also not displayed.



CPU ID: When more than one CPU is connected to the system through a communications network, each CPU must have a unique identifying number. The valid range is 0 to 90 for master-slave communications, or 0 to 255 for peer-to-peer. Although 0 can be entered, it is not a valid ID for communications. ID 255 is used to broadcast to all CPUs, and can be used if the actual CPU ID is not known.

Checksum: This entry appears only in On-line or Monitor mode with an operating CPU connected. The Series Six PLC automatically calculates a checksum for the ladder logic program during successive sweeps. If a checksum fault is detected (indicating a possible change in the ladder logic program) when the end of the program is reached in On-line or Monitor mode, the message "LOGIC MEMORY CHECKSUM ERROR" is displayed.

Memory Size: This is the number of 16-bit words of total logic memory available. The abbreviation K stands for 1024. This value is fixed by the hardware if the system is in On-Line or Monitor mode. In Off-Line mode, only this value can be changed.

Words Used: This is the length of the current program in 16-bit words.

CPU Memory: This entry is only displayed in On-Line or Monitor mode with a CPU connected. It shows the state of the CPU Memory Protect switch.

Words Available: This is the exact number of 16-bit words remaining for program storage. "Words Used" plus "Words Available" equals "Memory Size."

Subroutines Used: This is the number of subroutines that have been used in the program. (Range = 0 to 16) If more than 16 subroutines are placed in a program, an error message will appear while storing the program to a CPU or disk.

Registers: This is the number of 16-bit storage locations (registers). In On-Line or Monitor mode, when connected to a CPU, this is the total number of registers available. The CPU controls this entry. In Off-Line mode, this entry can be changed, as described in this chapter. It initially defaults to 16K.

CPU Status: In On-Line or Monitor mode with a CPU connected, this entry displays the state of the CPU.

CPU Mode	Description
Run	Scanning logic.
Enable	Hardware outputs in the I/O structure turn on or off, as dictated by the ladder diagram program.
Disable	Outputs are disabled, forced to the off state regardless of the program.
Stop	Halted.

Programmer ID: The programmer software:

Version	ID
LM6 prior to 4.0	225
LM6 4.0 parallel	000
LM6 FFP	001
LM6 4.0 serial	002

Functions: This entry identifies the level of features available in the CPU.

Function Level	CPU Microcode Version
Basic/1.0	8
Basic/1.01	9
Extended/2.0	101
Extended/2.01	102
Extended/2.02	103
Advanced/3.0	104
Advanced/3.01	105
Advanced/3.02	106
Expanded/4.0	110
Expanded/4.10	120
Expanded II/5.0	130
Expanded II/5.03	133

If the system is powered up in Off-Line mode, this defaults to Expanded II (version 5.0). Higher-level features are available only when the appropriate function level is selected. Do *not* change the function level displayed in the Scratch Pad with a file name active. This will result in the message "INVALID LAD FILE DETECTED" being displayed. Set the function level before naming and creating the ladder diagram program.

CPU Error Flags: This entry only appears in On-line or Monitor mode with a CPU connected. It displays the CPU errors encountered since the last time these errors were cleared.

CPU error flags are stored in Scratch Pad memory. The CPU uses these flags to record faults encountered during normal operation, or during self-checks. This display is always controlled by the CPU.

If any of the error flags is set, a message appears on the line below the error flags line. Some messages also show a hexadecimal address. This address (shown in the examples below as HHHH) may indicate that a module needs to be replaced. Example messages are:

OVERRIDE TABLE PARITY ERROR HHHH
I/O STATUS TABLE PARITY ERROR HHHH
TRANSITION TABLE PARITY ERROR HHHH
SCRATCH PAD PARITY ERROR HHHH
REGISTER MEMORY PARITY ERROR HHHH
LOGIC MEMORY PARITY ERROR HHHH
I/O CHAIN PARITY ERROR HHHH
I/O CHAIN CONTINUITY ERROR
DATA PROCESSOR ERROR
SERIAL COMMUNICATIONS INTERFACE ERROR

For CPU version 110, if an I/O chain parity error is encountered, the error message also displays a channel number in hexadecimal before the hex address.

Editing the Scratch Pad Display

The contents of the Scratch Pad can be changed at any time, with the following restrictions:

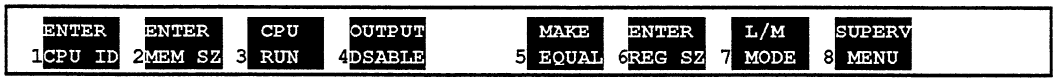
- In On-Line or Monitor mode, the CPU controls the values for memory size, run status, functions, registers, version, and error flags.
- In Off-Line mode, if there is a program loaded into system memory, changes are limited to those compatible with the program.

When attempting to store a program into the CPU, the system compares the values that are stored in its Scratch Pad memory with those in the CPU. It uses these values to ensure that the CPU has the capacity to run the program.

1. First, the system compares the CPU logic memory to the Scratch Pad entry for memory size.
2. If the CPU memory is less than the Scratch Pad entry, the system compares the actual size of the program to the available CPU program memory.
3. The system compares the Scratch Pad entry for registers with the available CPU registers, and if the CPU register size is less than the Scratch Pad entry, the system compares the actual size of the registers in the program to the available CPU register size.
4. The system compares the Scratch Pad entry for functions with the available CPU function level.
5. If the CPU function level is lower than the Scratch Pad entry, the system checks the program to see if the functions it uses are within the level of the CPU.

The program cannot be stored if any of these comparisons show that it is not suitable for use in the CPU.

The function keys shown below are used to change the Scratch Pad. They are displayed only when their functions can be used.



CPU ID

Each CPU must have a different ID. To change the entry for CPU ID, enter a new value on the bottom line of the work area. Enter 1 to 90 for master-slave communications, or 0 to 255 for peer-to-peer communications. If the CPU ID is not known, in Off-Line mode, enter 255. Then, press F1 (Enter CPU ID). After establishing communications, switch to On-Line mode. The actual CPU ID will be displayed. Then, the Scratch Pad entry for CPU ID can be changed to match the displayed ID.

Scratch Pad can also be used to change the ID in the CPU. If the system is on-line and connected to the CPU when this entry is changed, the CPU will assume the new ID.

CAUTION

(Serial versions only) For multi-drop communications, use care when making on-line changes to the CPU ID. Some versions of the CCM card may lose communications if the CPU ID is changed on-line.

Memory Size

In On-Line or Monitor mode, the entry for program memory size is set by the CPU. In Off-Line mode, you can change this entry by typing a new value on the bottom line of the work area and pressing F2 (Enter Memory Size). The new value must be a multiple of 2 (2 ... 32, or 64), but not the number 30. The entry represents K (1024) of memory words. If there is a program in system memory, the value entered here must be equal or greater than the size of the program. For example, if the program in memory uses 9217 words, the value 8 would not be accepted.

Registers

In On-Line or Monitor mode, this value is set by the CPU. To change the entry for registers, in Off-Line mode, enter a new value on the bottom line of the work area. Valid entries are 256, either 1024 or 1 (for 1K), either 8192 or 8, or either 16,384 or 16. Press F6 (Enter Register Size). If there is a program in system memory, the value entered here must be equal or greater. If the ladder logic includes a CPU Configuration instruction, the Scratch Pad entry for registers cannot be decreased.

CPU Status

CPU status can only be changed in On-Line mode, with a CPU connected. The CPU Run keyswitch must be in the ON position. Press F3 (CPU Stop) to toggle the CPU status between STOP and RUN (scanning logic).

Outputs Enable

This function can be used to force the state of all outputs in the system. "Outputs Enable" turns the hardware outputs ON or OFF, as dictated by the ladder diagram program. "Outputs Disable" forces all hardware outputs to their reset conditions, regardless of the program.

In On-Line mode with a CPU connected (the CPU Run keyswitch must be in the ON position):

- If "Start" and "Enable" are displayed, pressing Start changes the CPU status to "Run/Enabled."
- If "Start" and "Disable" are displayed, pressing Start changes the CPU status to "Run/Disabled."
- If "Stop" is displayed, pressing Stop displays the CPU status as "Stopped."

Function Level

In On-Line or Monitor mode, the CPU controls the Scratch Pad Display entry for CPU functions. In Off-Line mode, you can change this level to one that is compatible with the functions used by the program.

NOTE

In order to use the Expanded II function set, the CPU must have an Expanded II Logic Control Module (IC600CB515). Do *not* select the Expanded II function set if the program will be run on a CPU without this module.

1. Enter the new level into the top line of the work area. Type the number that represents the function set: 1 (Basic), 2 (Extended), 3 (Advanced), 4 (Expanded), or 5 (Expanded II).
2. Press F5 (Enter Function Set). If there is a program loaded into system memory that requires a higher level of functions than the one indicated, the change will not be accepted.
3. On returning to On-Line or Monitor mode, if the memory size, registers, CPU ID, and function level of the CPU are not the same as the new level, both levels are displayed on the Scratch Pad Display screen, with the CPU values in reverse video.
4. To make these values equal, press F5 (Make Equal). This key is only functional when there is a difference between the CPU and the Scratch Pad. The system will attempt to make its values equal to those of the CPU. If that is not compatible with the program, an error message will be displayed.

Chapter 4
Display Program

The Display Program function is used to display ladder logic, showing power flow through the rungs. If the program is annotated, names and nicknames can be shown in the program. Rung explanations and coil labels can be shown in Window mode, or in full-screen Page mode.

Display Program includes full search capabilities, allowing you to quickly locate any rung, reference, nickname, or element in the program. In addition, the Display Program function allows single-word changes when the system is on-line to an operating CPU. References and some instructions can be changed, relays overridden, and new data values entered.

Displaying a Program

The first time you use Release 3.01 or later software to display an older program, Logicmaster 6 converts the structure of the .NAM, .EXP, and .RDF files, making them incompatible with earlier versions of the software.

You should first copy these files onto a newly-formatted program diskette, and use these copies. If a disk error occurs while the files are being converted for display, copy only the .NAM, .EXP, and .RDF files from the diskette and try again.

To display a program:

1. Display Program displays the program currently in system RAM memory. If the program you want to display is the one now in memory, go to step 5.
2. If the program is not currently in RAM memory, you can load it using the Load/Store/Verify functions, or the quick-load feature from the Supervisor menu. For on-line display, the program in the CPU and the program in memory must be equal. If the program is loaded from the CPU, they are equal. Otherwise, the program in memory must be stored to the CPU.
3. To load a program from the CPU, or check the accuracy of the transferred program using the Verify function, go to step 4. Otherwise, you can quickly load the program directly from the Supervisor menu, as described in this step.
   A. If the file name is not the active file name, type it in. A drive ID followed by a colon can be used before the program name if the program is not on the default disk. Then, press the Enter key.
   B. When the file name is active, press ALT-L to load the file.
   C. Go to step 5.
4. To load a program from the CPU, or use the Verify function, select Load/Store/Verify (F6) from the Supervisor menu.
   A. Press F1 (Load) to display the Load Program/Tables screen.
   B. Enter the drive ID for the program file. This may be P for the CPU, or it may be the diskette or hard disk ID.
   C. After entering the drive ID, move the cursor to Program Name and enter the name of the program to be displayed. Then, press the Enter key.

D. When the Load is complete, use the Verify function to check the program content. For more information, refer to chapter 9, *Load/Store/Verify*.

E. Return to the Supervisor menu through the Load/Store/Verify menu, or by pressing the Supervisor key.

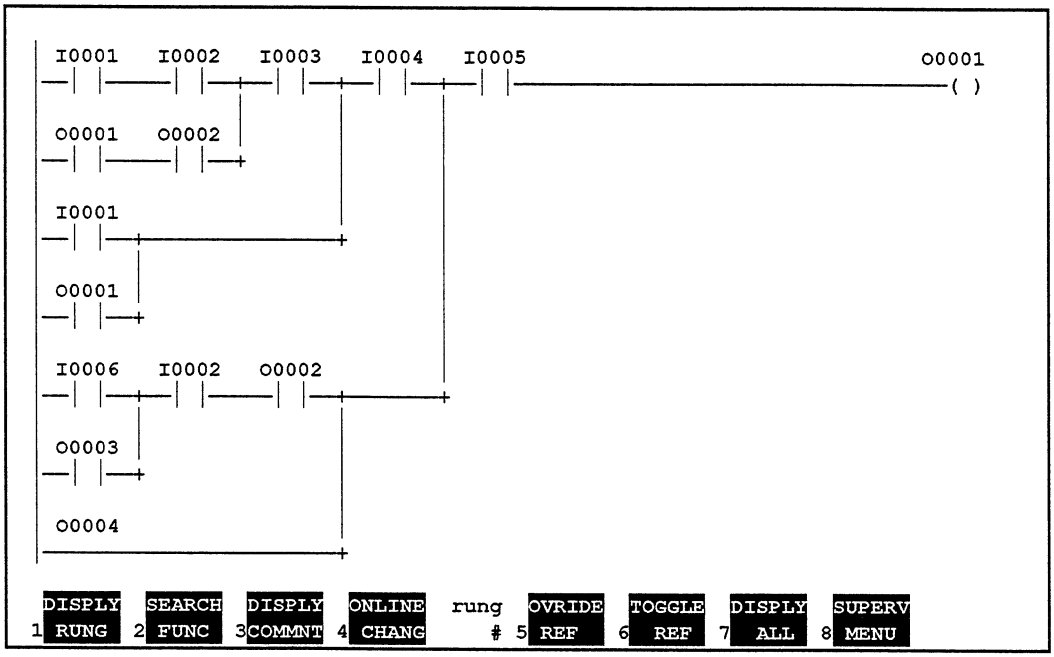
5. In the Supervisor menu, type in the name of the program to be displayed and press F1 (Display Program).

NOTE

If there is no name in the text area, the last program name used in either Display Program or Edit Program is used. If no program name is either entered or retained from the Display or Edit Program functions, the system displays the last program that was used. You can delete this name by typing in NONE.

Program Display Format

When the program is first displayed, it is either at the area last displayed in the Edit or Display Program mode, or at the beginning of the program if it is a new program or if memory was reloaded.



The program is displayed as lines of relay logic, including power flow of relay contacts. Reverse video appears at those relay contacts that transmit power flow from the left to the right. If the system is on-line to an operating CPU, the current values of references appear.

Up to eight lines can be connected to a coil to form a rung of logic. The screen displays seven lines at one time. The reference, element type, and value can be shown. If nicknames have been used in the program, they will appear over the appropriate elements, in place of the reference addresses.

The bottom of the Display Program screen shows the key functions that are currently available. For example, if the cursor is at [Start of Program], the functions of Override Reference and Toggle Status cannot be used. Therefore, keys F5 and F6 display no function assignments. If you move the cursor

down the program display, you will see the available key functions change. The functions appropriate to the current cursor location are the ones which are displayed.

At the initial Display Program menu, function keys F1 through F8 have the following key assignments:

Function Key	Function	Description
F1	Display Rung	Select another rung for display.
F2	Search Function	Search for a program element.
F3	Display Comment	Select Window or Page mode. Use this key in conjunction with Display All/Display Nicknames (F7) to select the format for annotation display.
F4	On-Line Change	Make single-word changes to the program. This function is available only when on-line to an operating CPU, with the displayed program equal to the program executing in the CPU. For serial communication, the On-Line Change field in the Expanded Functions menu must be set to ENABLED. Refer to chapter 10, *Expanded Functions Menu*, for more information.
F5	Override Reference	Toggle the override status of the input, output, auxiliary input, or auxiliary output at the cursor position. This function is only available when on-line to an operating CPU.
F6	Toggle Reference	Toggle the status of the input or output reference at the cursor position. If the reference is currently overridden, the new status is retained until F6 is pressed again. If the reference is not currently overridden, the new status is retained only until changed by the program.
F7	Display All	Toggle the format of the ladder logic display between references/nicknames display and annotated display. In Display All mode, reference addresses, nicknames, and names appear over each element. The .NAM file for the program must be present in the default drive for proper annotated display. Refer to chapter 6, *Annotation*, for more information.
F8	Supervisor Menu	Return to the Supervisor menu.

Fast Update - Serial Versions

Serial versions of the software provide fast update of a selected reference during Display Program mode. The Fast Update function updates the selected reference very rapidly, while other references are updated more slowly.

To use the Fast Update function, the system must be in On-Line mode with CPU communications active, and the program in memory must be equal to the program in the CPU. Move the cursor to the rung element to be updated, and press ALT-U. The screen displays the message "Fast Update Mode."

Fast Update will continue until any other key is pressed, the keyswitch is turned off, or the program in memory and the program in the CPU are no longer equal.

Selecting a Rung for Display

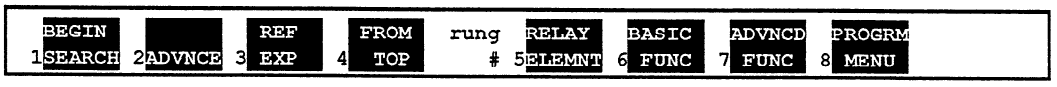
To display another rung at the top of the screen:

1. Enter one of the following parameters.
   A. A rung number. Enter a decimal, unsigned value on the bottom line of the screen's work area.
   B. A memory address. Enter a hexadecimal value on the bottom line of the screen's work area.
   C. A coil reference. Enter an appropriate O, AO, or Expanded Output value on the center line of the work area.
   D. A nickname. Enter a nickname on the top line of the work area.
2. Press F1 (Display Rung).

The rung will be displayed at the top of the screen. The cursor will be positioned at the first element, or memory address, or coil - depending on which was selected in step 1. If the selection is by memory address to a word that is not displayable, the cursor will be placed adjacent to the requested address.

Searching for a Program Element

To search for an element of a program being displayed on the screen, press F2 (Search Function). The following function key assignments are available:



Function Key	Function	Description
F1	Begin Search	Begin searching for the element defined in the work area.
F2	Advance	Toggle the search direction between forward and reverse.
F3	Reference Explicit	Toggle the search between explicit (REF EXP) and implicit (REF IMP) references.
F4	From Top	Toggle the starting point for the search between the top of the program and the current cursor location.
F5	Relay Elements	Display the Relay Element Search function keys.
F6	Basic Functions	Display additional Search keys: Shift/Move (F3) Basic Arithmetic (F4) Special Functions (F5)
F7	Advanced Functions	Display additional Search keys: Data Move (F1) Expanded Arithmetic (F2) Search Table (F3) List Functions (F4) Matrix Functions (F5) Control Functions (F6)
F8	Program Menu	Return to the Program menu.

1. To search for a reference or nickname, enter the reference or nickname in the work area. To search for a specific instruction, use the Search function keys. If an instruction and a nickname or reference are entered together, the combination is searched for.

   An asterisk (*) can be used to search for similar nicknames. It must be used at the end of the nickname, to replace the variable characters at the end of the nickname. For example, entering PB* would search for PB1, PB2, or PB3. This is known as a "wildcard" nickname search.

2. After you identify the element to be searched for, select the method of searching:

   A. Press F2 (Advance) to toggle the search direction between forward and backward.

   B. Press F3 (Explicit Reference) to toggle between:

      (1) Explicit References only - to search only for references actually entered in the ladder diagram and displayed on the screen.

      (2) Explicit and Implied - to search for both explicit and implied references, which are not displayed, but which are included in tables, matrixes, and similar functions.

   C. Press F4 (From Top) to toggle the starting place for the search between the top of the ladder diagram and the current rung.

3. Press F1 (Begin Search) to begin searching. When the element is located, that element will be displayed at the top of the screen.

4. To repeat the search, adjust the parameters as needed and press F1 again.

5. To exit from the Search function, press F8 (Program Menu). The system returns to the Display Program function.

Searching for the Cause of a Double Left Rail

A double left rail in the ladder logic means a Master Control Relay or Skip is controlling execution of the rungs. To locate the cause of a double left rail, search backward from the current rung. Search first for an MCR, then for a Skip if necessary. Do not include references.

Making On-Line Changes

Single word changes can be made in Display Program mode. The effect of on-line changes depends upon the current mode (On-Line, Off-Line, or Monitor), and whether the program in the CPU is equal or not equal to the program in Logicmaster 6 system memory. For serial versions of the software, it also depends on whether on-line changes are enabled.

When the system is in On-Line or Monitor mode with an operating CPU connected, the information displayed is obtained from the CPU. This information includes I/O state, override status, and register content. Displays are updated as the CPU status changes.

When connected to a scanning CPU, with the program in memory equal to the program in the CPU, in the On-Line or Monitor mode, power flow to the rungs of the ladder diagram is shown in reverse video. As the CPU status changes, displays are updated. If rungs are added or deleted in Edit mode, the program in memory and the CPU become "Not Equal", and displays are no longer updated.

When the system is in Off-Line mode, the status is always obtained from internal system memory. Power flow is shown with numerical values from internal memory.

WARNING

Improper use of on-line program changes can damage equipment or cause personal injury. Make on-line program changes with extreme care. On-line program changes, overriding references, and forcing references can have serious and unforeseen results on a control system if they are improperly used. These functions should not be used with people near the equipment. If possible, they should be done with direct visual control over the system. Proper external power disconnects should be made to prevent undesired equipment operation.

To make an on-line single word change:

1. The status line at the top of the screen must show that the system is on-line to the CPU, and that the active program is exactly the same (''equal''). For example:

```
CPU: RUN/ENABLE   SWEEP:  7ms   L/M EQUAL CPU   L/M: ONLINE   CURSOR: 000C
```

2. Display the rung that contains the element to be changed. If the change is to a reference address or value, go to step 3.
3. To change an instruction, press F4 (On-Line Change). The bottom of the screen displays the new function key assignments.
4. Place the cursor on the element to be changed.
   A. Numeric values may be displayed in decimal, signed decimal, or hexadecimal format. The base displayed depends on the type of function that uses the reference. For instructions that permit it, the base can be changed by placing the cursor on the reference to be changed, and pressing the Shift key at the same time as the appropriate Display Mode key.
   B. To change a reference only, type the new reference or its nickname into the work area and press the Enter key. Pressing Shift-Enter changes both the reference and the data at the same time.
   C. To change the element type, press the appropriate function key displayed at the bottom of the screen. The function key assignments vary depending on the location of the cursor. You can change the element type as follows:
      - Contact - Reverse contact type: NO to NC, or NC to NO.
      - Coil - Reverse coil type between relay and one-shot.
      - Counter - Change counter type, or replace counter with timer.
      - Timer- Change timer type, or replace timer with counter.
   D. To change the content of a register, or 8 or 16 bits of I/O, place the cursor on the item to be changed. Then, type the new value into the work area and press the Enter key.

Making On-Line Changes, Serial Versions

For serial versions of the software, on-line changes must be enabled in the Communications Setup menu. If they are not set up, the function key assignment will not be available.

When making single-bit changes, the system reads the value of the bit to be toggled on one sweep of the CPU. It writes back the command to make the change on a later sweep. It is possible for the program or an external device to change the value of the bit before the Write command is received. If so, the value will then be incorrect.

Occasionally, with all types of on-line changes (bit, byte, or word) errors may occur because the system does not verify the change until a later sweep. If an error has occurred, the system cannot try the change again. This is less likely to occur if the CPU is stopped temporarily to make the change.

NOTE

If the version of the CCM card in the CPU is earlier than version 5, program memory (not tables or registers) cannot be written to while the CPU is running.

If the version of the CCM card is earlier than version 5, and the CPU was programmed using SCREQ commands, the system may encounter write-protected memory. If so, the changes will not be permitted.

Overriding an Input or Coil

You can override the status of each input and each coil in the Main I/O and Auxiliary I/O status tables. An override removes control of the reference from its normal source. Overridden inputs ignore information from the devices wired to the I/O structure, such as limit switches or pushbuttons. Similarly, overridden outputs ignore programmed logic and internal power flow. Non-relay functions such as timers, counters, arithmetic functions, and moves still function when a coil is overridden.

When a reference is overridden, all occurrences of that reference in the program will remain in either their ON or OFF state. References that were off when overridden will remain off; references that were on will remain on. These changes will occur throughout the program. Overrides are retained even when power is removed from the system.

The ladder diagram logic cannot change overrides; however, non-relay functions such as timers, counters, arithmetic functions, and moves can change the state of an overridden reference.

It is possible to override all inputs, outputs, and auxiliary I/O references. Expanded references in channels 1-7 and 9-F cannot be overridden.

The Override is a very powerful tool for program checkout and for maintenance. You can test a program in a PLC that is not connected to an I/O structure by using overrides to simulate inputs. You can also check out a program when I/O is connected, by using overrides to prevent coil operation.

After the I/O is wired up, it can be tested by activating each coil with an override to verify I/O communications, module addressing, module operation, power to a device, wiring to a device, indicator lights, fuses, and other hardware.

After the control system is thoroughly checked out and placed in operation, the override is very useful in a monitored system. If a sensor or input module should fail while the process is in operation, that input can be overridden. Thus, the process can be continued until it can be shut down safely.

Identifying Overrides during Display Program

In Display Program mode, when a reference is overridden, the first character of its nickname blinks. If annotation is not used in the program, then the I or O of the reference address blinks.

Using the Display Reference Function to Identify Overrides

References should not be overridden when the Logicmaster 6 system is removed from the process, or when making copies of the program. Use the Display Reference function to verify all inputs and coils before removing the programming device, or copying the program.

Searching for an Active Override

The CPU uses outputs O1015 and O1016 to find and indicate active overrides. If O1015 is set to 1, the CPU searches the override tables. If any active override is found, the CPU sets bit O1016 to 1. Checking O1016 will show whether there are currently any active overrides.

Using Overrides

Overrides should be used on an operating system only with extreme care.

WARNING

Improper use of the override can damage equipment or cause personal injury.

To use an override, the CPU memory protect switch must be in the OFF position.

1. Display the area of the program you want to change and press the On-Line Change (F4) key. The screen shows new function key assignments.
2. Place the cursor on the reference to be overridden.

CAUTION

The reference will be overridden throughout the program, not just at the cursor location.

3. Press F4 (Override). This key toggles the state of the reference between overridden and not overridden.

Forcing the Status of a Reference

In the Display Program function, the Toggle Reference (F6) key is used to toggle the status of any reference in the program between ON and OFF. If the reference that is toggled is currently overridden, it retains its new status until the Toggle Reference key is pressed again. If the reference is not overridden, it retains its new status until changed by some other function, such as rung solution or I/O servicing. This usually occurs within one scan.

To toggle a reference with the program displayed on the screen, place the cursor on the element whose reference is to be forced. Press F6 (Toggle Reference) to change that reference to its opposite state. All logic elements in the program that use that reference will reflect the new status.

Chapter 5
Edit Program

GEK-25379

The Edit Program function is used to create or modify a ladder logic program in system memory.

If the Edit function is entered with an active file name, annotation can be edited. In addition, disk memory will be updated whenever a rung is accepted. A file name is activated from the Supervisor menu by entering a valid file name and pressing the Enter key or the Edit Program function key.

If a file name is not active, the program must be stored to disk using the L/S/V functions. That means if power goes off while editing, changes will be lost.

The Edit Program function is available in all modes. However, register contents and I/O numeric values are shown in Off-Line mode only.

Like Display Program, the Edit Program function contains full search capabilities. You can quickly locate any rung, reference, nickname, or other element in a program.

This chapter explains how to use the Logicmaster 6 system to enter or modify a ladder logic program. Refer to the following sections.

Section 1. Entering Edit Program Mode: explains how to begin the Edit Program function.

Section 2. Editing the Program: explains how to edit a program.

Section 3. Editing a Rung: explains how to create or edit a rung.

Section 4. Searching for a Program Element: explains how to search for a particular element of a program, such as a reference or a program instruction.

Section 5. Ladder Diagram File Editing: explains how to copy part of a program into another file and how to merge program files.

Selecting Program Windowing

"Windowing" is used to create programs with more than 6,000 rungs and/or more than 2K nicknames. Windowing allows up to 10,000 rungs and up to 5K nicknames. Windowing can also be used to enable ladder logic programs larger than 32K words.

Before creating a program or editing an existing program, you should be familiar with the use of windowing. This is particularly true if you are loading a program for editing which has more than 6,000 rungs or 2K nicknames. For more information, refer to chapter 10, *Expanded Functions Menu.*

SECTION 1
Entering Edit Program Mode

Starting a New Program

To start a new program:

1. Edit Program edits the program in system RAM memory. If there is another program already in RAM memory, you must either store it or clear memory using the Load/Store/Verify functions. If necessary, refer to chapter 9, *Load/Store/Verify*, for instructions.
2. When memory is clear, you can begin the new program. In the work area:

   A. Type the program name, if you want to automatically update the program on disk after each rung is accepted. This is recommended. Entering the program name is also necessary for annotation.

   The name can have up to 8 characters. It should be different from any other program name in the system. Uppercase and lowercase characters are displayed, but are considered to be the same by the system. Do not use the following reserved words as program names:

   ```
   NONE, CON, PRN, AUX, COM1, COM2, LPT1, LPT2, LPT3, NUL
   ```

   Press the Enter key to enter the name.

   B. Type the word NONE, if you do not want to enter a name. To save the program later, you will have to store it to disk using the Load/Store/Verify functions.
3. Press the Edit Program (F2) key. The beginning of a new program appears. Now go to section 2, "Editing the Program."

Displaying an Existing Program for Editing

The first time you use Release 3.01 or later software to edit an older program, Logicmaster 6 software converts the structure of the .NAM, .EXP, and .RDF files, making them incompatible with earlier versions of the software.

You should first copy these files onto a newly-formatted program diskette, and use these copies. If a disk error occurs while the files are being converted for editing, copy only the .NAM, .EXP, and .RDF files from the diskette and try again.

To edit an existing program:

1. Edit Program edits the program currently in system RAM memory. If the program you want to edit is the one now in memory, go to step 5.

2. If a program must be loaded from disk or from the CPU, as described in steps 3 and 4, be sure to select windowing if the program may exceed 6000 rungs, 2K nicknames, or 32K words. Instructions for selecting windowing are given in chapter 10, *Expanded Functions Menu.*

NOTE

The Logicmaster 6 software must be re-started after changing the selection for windowing. Selection of windowing is stored in a file named MACHINE.SET. This file is only read at startup.

3. If the program is not currently in RAM memory, you can load it using the Load/Store/Verify functions, or the quick-load feature from the Supervisor menu. For on-line edit, the program in the CPU and the program in memory must be equal. If the program is loaded from the CPU, they are equal. Otherwise, the program in memory must be stored to the CPU.
4. To load a program from the CPU, or check the accuracy of the transferred program using the Verify function, go to step 4. Otherwise, you can quickly load the program directly from the Supervisor menu, as described in this step.
   A. If the file name is not the active file name, type it in. A drive ID followed by a colon can be used before the program name if the program is not on the default disk. Then, press the Enter key.
   B. When the file name is active, press ALT-L to load the file.
   C. Go to step 5.
5. To load a program from the CPU, or use the Verify function, select Load/Store/Verify (F6) from the Supervisor menu.
   A. Press F1 (Load) to display the Load Program/Tables screen.
   B. Enter the drive ID for the program file. This may be P for the CPU, or it may be the diskette or hard disk ID.
   C. After entering the drive ID, move the cursor to Program Name and enter the name of the program to be displayed. Then, press the Enter key.
   D. When the Load is complete, use the Verify function to check the program content. For more information, refer to chapter 9, *Load/Store/Verify.*
   E. Return to the Supervisor menu through the Load/Store/Verify menu, or by pressing the Supervisor key.
6. In the Supervisor menu, type in either:
   A. The program name, if you want to automatically save the program to disk after each rung is accepted. This is recommended. Entering the program name is also necessary for annotation.
   B. The word NONE, if you do not want to enter a name. To save the program later, you will have to store it to disk using the Load/Store/Verify functions.
7. Press the Edit Program (F2) key. If a program with the name entered is located on the disk, the system checks the contents of its internal memory against the contents of the disk. Any differences are noted. However, the system memory is not loaded from the disk at this time.

If no program is found with the name that you entered, no comparison is performed. A new editable copy of the program is stored to the active disk under the specified file name.

NOTE

If you load an existing program but give it a new name in the work area, an editable copy of the program will be stored to disk under the new name. The new name must not be the same as the name of any other program on the disk.

Creating a Backup Program

A backup program is a copy of the program before any new editing changes are made. Only one backup version of a program is permitted. A new backup destroys any old backup versions of the same program.

When you enter Edit Program mode, if there is a program already in memory with the same name, the screen prompts:

```
DO YOU WISH TO BACKUP PROGRAM? (Y/N)
```

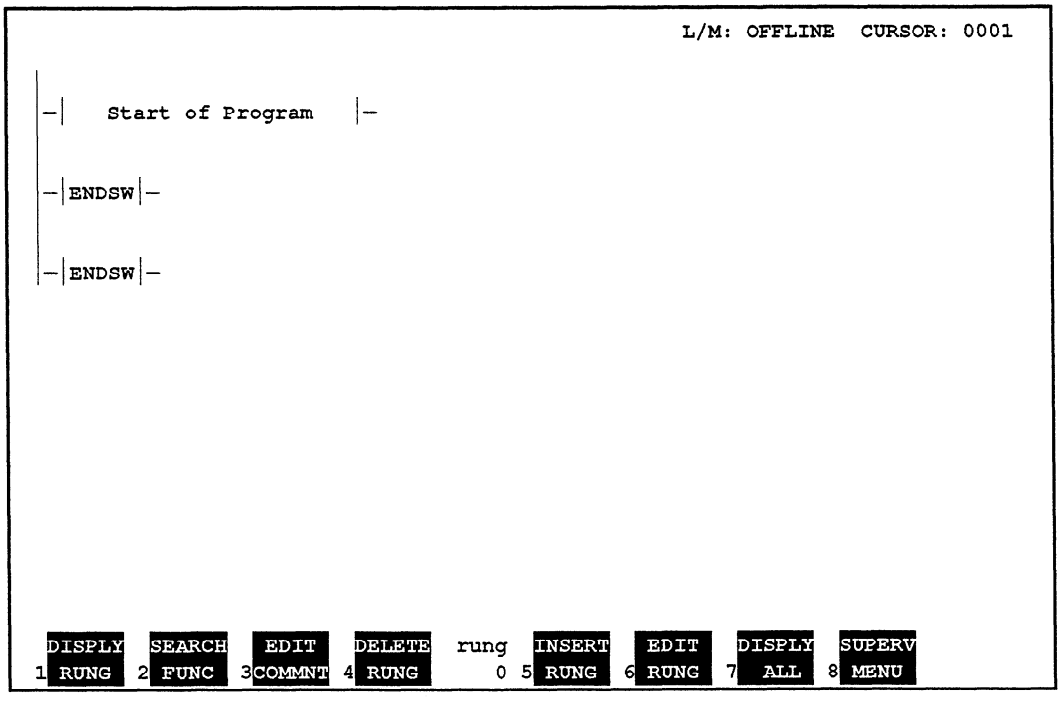
To make a backup program, type **Y**. A copy of the current version will be stored on the disk. If you do not want to back up the program, type **N**.

NOTE

In addition to making a backup copy of the file on disk, you should also keep a library of backup diskettes.

SECTION 2
Editing the Program

When the Edit Program function is selected from the Supervisor menu, the program is displayed on the screen. If it is a new program, as shown below, it has a Start of Program rung (rung 0), and two End of Sweep elements. These are the minimum program content. They cannot be deleted.



The bottom of the Edit Program screen shows the function key assignments and the current rung location of the cursor. The key assignments are only available where they are permissible. For example, if you did not enter a program name, the Display All (F7) function is not displayed.

Function Key	Function	Description
F1	Display Rung	Display the rung number or output coil entered in the work area.
F2	Search Function	Search for a program element.
F3	Edit Comment	Edit annotation in Window or Page mode. Use this key in conjunction with Display All/Display Nicknames (F7) to select the format for annotation display. Refer to chapter 6, *Annotation*, for information on editing annotation.
F4	Delete Rung	Delete the rung at the cursor position. Rung explanations in the annotation file will be renumbered automatically if a file name is active.
F5	Insert Rung	Display the Edit Rung function keys, which are used to select program elements. Rung explanations in the annotation file will be renumbered automatically, if a file name is active.
F6	Edit Rung	Edit the rung at the cursor location. This function displays the Edit Rung keys, which are used to select the program elements.
F7	Display All	Toggle the format of the ladder logic display between reference/nickname display and annotated display. The .NAM file for the program must be present in the default drive for proper annotated display. Refer to chapter 6, *Annotation*, for more information.
F8	Supervisor Menu	Return to the Supervisor menu.

Displaying a Specified Rung

To display a specific rung:

1. Type one of the following into the work area:
   A. Enter the number of the rung on the data line, in decimal format.
   B. Enter the coil reference on the reference line.
   C. Enter the memory address on the data line, in hex format.
   D. Enter the nickname on the text line.
2. Press F1 (Display Rung). The selected rung appears at the top of the screen, followed by up to six lines of the program. The cursor indicates the rung, coil, or memory address that was specified.

Inserting a Rung

To insert a rung:

1. If the new location for the rung is not shown, display it using the Search or Display Rung function.
2. Place the cursor on an element in the rung *before* the location where you want the new rung.
3. Press F5 (Insert Rung).

Editing a Rung

To edit a rung:

1. If the rung is not shown, display it using the Search or Display Rung function.
2. Place the cursor on the rung to be edited.
3. Press F6 (Edit Rung).
4. Continue at section 3, ''Editing a Rung''.

Deleting One or More Program Rungs

To delete one or more rungs:

1. Decide how many rungs will be deleted.
2. Place the cursor on the first rung to be deleted. If the rung is not shown on the screen, display it using the Search or Display Rung function.
3. If you are deleting just one rung, press F4 (Delete Rung). Go immediately to step 4.

   If you are deleting multiple rungs, use the Select key to move the work area cursor to the bottom line. Then type in the number of rungs to be deleted. Press the ALT-D keys to delete the rungs.
4. To confirm the deletion, press the Confirm (Shift-0) key. To cancel the deletion, press any other key.

Reference Substitution

Reference Substitition allows you to replace all occurrences of an explicit reference in a program. This lets you randomly select I/O and register points during program development. After the real hardware points are established, the Reference Substitution function is used to replace the randomly-selected points with correct reference types and addresses.

The reference can be replaced in two ways. The first way allows you to replace a reference with one of the same type (for example, one register for another). The second way allows you to replace a reference with one of a different type (for example, an output for an input).

To replace a reference in a program:

1. If the program is not currently in memory, load it as previously described. The program name must be the active file name.
2. With the Edit Program main menu keys displayed, enter the new reference on the reference (center) line of the work area.

   A. To replace the same type of reference, enter the old reference on the data (bottom) line of the work area. Enter only the reference number, not the reference type. The following entries would replace reference R0003 with R0143:

Text Line		(no entry)
Reference Line	R0143	NEW Reference
Data Line	0003	OLD Reference

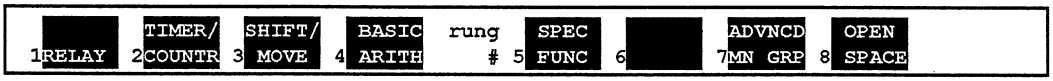
B. To replace a different type of reference, enter the assigned nickname of the old reference into the text (top) line of the work area. The reference must not already be assigned to another nickname. The following entries would replace the current reference assigned to nickname LIMIT-SW (for example, AO0053) with reference AI0017:

Text Line	LIMIT-SW	Nickname
Reference Line	AI0017	NEW Reference
Data Line		(no entry)

3. Press ALT-S to begin the replacement. If the replacement is appropriate, the reference will be changed throughout the program. Both the .LAD and .NAM files will be updated. If the replacement cannot be completed, you can stop it. The program will be restored to its original content.

SECTION 3
Editing a Rung

Selecting either Insert Rung (F5) or Edit Rung (F6) from the Edit Program screen displays the following function key assignments:



These keys are used to add elements to a rung. Their functions are listed later in this section. To edit a rung:

1. Place the cursor at the position for the new logic.
2. For a simple function such as a relay, you can enter the reference and then select the element. The element will appear with the reference above it.

For a function requiring more than one reference, select the function first. Then, enter the references as indicated by the display. For example, if you select the ADD function, the screen displays:

```
R****R**** R****
 -|  A+B=C  |-
```

Here, the display shows you to enter three register addresses, as indicated by the "R****" over each part of the function. As another example, if you are entering a timer with a constant preset, the screen displays:

```
  Const
-|PRESC|-
   ***
```

You would enter a constant value.

More information on entering values in the work area is given in chapter 2.

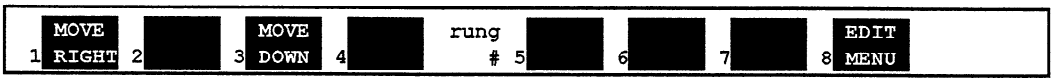
Entering a Reference

To enter a reference, type in the reference type and address using the numeric keypad. When the correct value has been entered, press the Enter key. (To enter both the reference and a value for the reference, enter the reference on the middle line of the work area, and enter the value on the lower line of the work area. Then, press the Shift and Enter keys at the same time.

Many functions require references that begin on a byte boundary. If a value is entered which is not byte aligned, the system adjusts the entry to be on a byte boundary. The screen displays the new value assigned to the reference.

Adding an Open Space to a Rung

The Open Space (F8) function opens element spaces in a rung. Use the F1 and F3 function keys to select the direction.



Function Key	Function	Description
F1	Move Right	All elements to the right of the cursor and the element at the cursor position move one position to the right in that line only. A new element can now be inserted at the cursor position.
F3	Move Down	Open a new rung line. By using this key, space for a new parallel contact can be inserted between two parallel contacts. Unused spaces will be deleted when the rung is accepted. This key assignment is only displayed when the cursor is not on the top line of a rung.
F8	Edit Menu	Return to the basic Edit function key assignments.

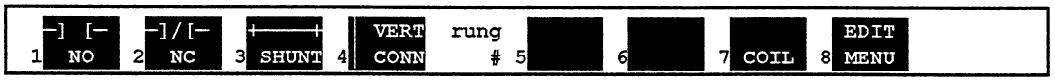
Exiting a Rung

While you are editing a rung, the cursor remains in that rung. There are two ways to leave the rung:

1. Complete the rung, and enter it into the program by pressing the Accept key. At this time, the system checks the construction of the rung. If the rung is accepted, the edit functions appear at the bottom of the screen. The cursor can now be moved to another rung.
2. To terminate editing the rung without placing any changes in the program, press the Abort key. Respond to the prompt that appears by pressing the Confirm (Shift-0) key. If any other key is pressed, that rung remains open for further editing and is not aborted.

Relay Functions

The Relay (F1) key displays the following function key assignments:



Function Key	Function	Description
F1	Normally Open Contact	Select a normally open contact.
F2	Normally Closed Contact	Select a normally closed contact.
F3	Horizontal Shunt	Select a horizontal shunt. A shunt is a connector with no contact function or reference.
	Horizontal Open Space	If there is an element at the cursor location, the function of F3 changes to Horizontal Open Space. It will then replace an element with an open space.
F4	Vertical Connector	Create a vertical connection from the left of a relay element to the next lower line of logic. This function is not valid in the first column of the ladder.
	Vertical Open Space	If there is a vertical connection at the cursor location, the function of F4 changes to Vertical Open Space. It will then replace a vertical connection with an open space.
F7	Coil	Move the cursor to the tenth (coil) column of the ladder. Horizontal shunts (+----+) are entered as needed.
F8	Edit Menu	Return to the basic Edit function key assignments.

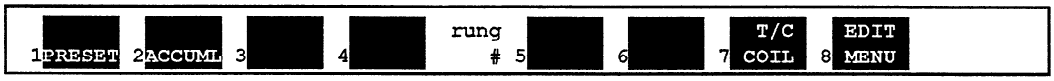
When F7 is pressed, these additional function key assignments are displayed:

Function Key	Function	Description
F5	Latch	Place a latch in column 9, a latch coil in column 10, and an unlatch coil in column 10 of the next available line. Used only when the cursor is in column 10.
F6	One-Shot	Place a one-shot coil in column 10. Used only when the cursor is in column 10.
F7	Relay	Place a relay coil in column 10.

For more information about these functions, refer to chapter 13, *Advanced Functions*.

Timer/Counter Functions

The Timer/Counter (F2) key displays the following function key assignments:



Preset (F1)

Press F1 to move the cursor to the ninth column of the top rung line, and display the following function key assignments:

Function Key	Function	Description
F1	Preset Constant	Place a fixed preset in column 9.
F2	Preset Register	Place a register preset in column 9.

Accumulate (F2)

Press F2 to move the cursor to the ninth column of the lower rung line, and display the following function key assignments:

Function Key	Function	Description
F1	Preset	Move the cursor up to the preset position, and display the Preset function keys.
F2	Accumulate Register	Place a register for storing the current timer/counter value in column 9 (lower line).

Timer/Counter Coil (F7)

Press F7 to move the cursor to the tenth column and display the following function key assignments:

Function Key	Function	Description
F3	Counter Up	Select an Up counter.
F4	Counter Down	Select a Down counter.
F5	Timer Seconds	Select a timer with a clock rate of one pulse per second.
F6	Timer 1/10 Sec	Select a timer with a clock rate of 10 pulses per second.
F7	Timer 1/100 Sec	Select a timer with a clock rate of 100 pulses per second.

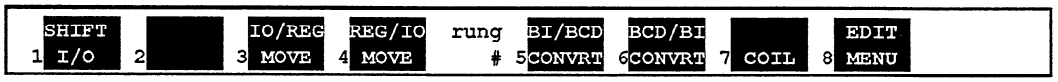
Edit (F8)

Press F8 to return to the basic Edit function key assignments.

For more information about these functions, refer to chapter 13, *Advanced Functions.*

Shift/Move Functions

The Shift Move (F3) function key displays the following function key assignments:

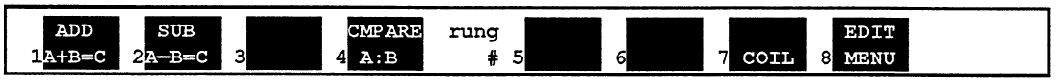


Function Key	Function	Description
F1	Shift I/O	Move 16 bits from a specified input or output address to the next higher address.
F3	I/O to Register Move	Move 16 input or output bits into a register. Not shown in the ninth column.
F4	Register to I/O Move	Move the content of a register into 16 input or output bits. Not shown in the ninth column.
F5	Binary to BCD	Convert the binary value in a register to a BCD value, which is placed in the I/O tables. Not shown in the ninth column.
F6	BCD to Binary	Convert a BCD value from the I/O tables to a binary value, which is placed in a register. Not shown in the ninth column.
F7	Coil	Move the cursor to the tenth (coil) column, entering horizontal shunts as necessary. Displays one-shot and relay coil labels.
F8	Edit Menu	Return to the basic Edit function key assignments.

For more information about these functions, refer to chapter 13, *Advanced Functions.*

Basic Arithmetic Functions

The Basic Arithmetic (F4) function key displays the following function key assignments:



Function Key	Function	Description
F1	Addition	Add the contents of two registers and places the result in a third register. Not shown in the eighth or ninth column.
F2	Subtraction	Subtract the value in one register from the value in another, and places the absolute value or the result in a third register. Not shown in the eighth or ninth column.
F4	Compare	Compare the values in two registers. Not shown in the ninth column.
F7	Coil	Move the cursor to the tenth (coil) column, entering horizontal shunts as necessary. Displays one-shot and relay coil labels.
F8	Edit Menu	Return to the basic Edit function key assignments.

For more information about these functions, refer to chapter 13, *Advanced Functions.*

Special Arithmetic Functions

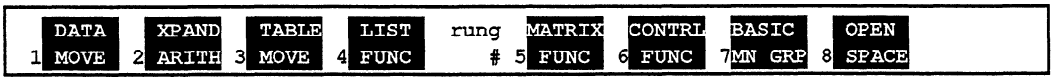
The Special Arithmetic (F5) function key displays the following function key assignments:

Function Key	Function	Description
F1	Master Control Relay	Cause the specified number of rungs after the MCR to be skipped. Output coils within the skipped area are turned off, unless overridden.
F2	Skip	Cause the specified number of rungs after the Skip to be skipped. Output coils within the skipped area are not turned off, but remain in the same condition as before the Skip.
F4	No Operation	Select the No Operation function.
F5	Serial Communications Request	Control serial communications with the CCM card of the Series Six programmable controller.
F6	Data Processing Request	Communicate with the DPU or ASCII/BASIC option of the Series Six programmable controller.
F7	Coil	Move the cursor to the tenth (coil) column, entering horizontal shunts as necessary. Displays one-shot and relay coil labels.
F8	Edit Menu	Return to the basic Edit function key assignments.

For more information about these functions, refer to chapter 13, *Advanced Functions.*

Advanced Functions

The Advanced Functions (F7) function key displays the following function key assignments, if Scratch Pad is configured for Advanced, Extended, or Expanded functions.



Data Move (F1)

Press F1 to display the following function key assignments:

Function Key	Function	Description
F1	Move A to B	Move 16 bits of data from one location to another. Not shown in the ninth column.
F2	Move Left 8	Move the left 8 bits from one 16-bit storage location to another. Not shown in the ninth column.
F3	Move Right	Move the right 8 bits from one 16-bit storage location to another. Not shown in the ninth column.
F4	Block Move	Move up to 7 values from the program to a specified destination. Shown only in the first or second column.
F7	Coil	Move the cursor to the tenth (coil) column, entering horizontal shunts as necessary. Displays one-shot and relay coil labels.
F8	Edit Menu	Return to the basic Edit function key assignments.

Expanded Arithmetic (F2)

Expanded Arithmetic functions include signed and double precision arithmetic for Extended, Advanced, and Expanded level CPUs. Floating Point functions for Expanded level CPUs are also included. Press F2 to display the following function key assignments:

Function Key	Function	Description
F1	Signed Addition	Add the value in one reference to the value in another, and places the signed result in a third. Not shown in the eighth or ninth column.
F2	Signed Subtraction	Subtract the value in one reference from the value in another, and places the signed result in a third. Not shown in the eighth or ninth column.
F3	Signed Multiplication	Multiply the value in one reference by the value in another, and places the result in a third. Not shown in the eighth or ninth column.
F4	Signed Division	Divide the value in one reference by the value in another, and places the remainder and quotient in a third and fourth reference. Not shown in the eighth or ninth column.

Function Key	Function	Description
F5	Double Precision Arithmetic	Select the Double Precision Arithmetic function. These functions operate on 32 adjacent bits of data (or 2 registers). Press F1 to select a Double Precision Addition function. Shown only in the first four columns. Press F2 to select a Double Precision Subtraction function. Shown only in the first four columns. Press F3 to select a Greater Than function to compare two signed numerical values. Shown only in the first six columns.
F6	Floating Point Arithmetic	Floating Point Arithmetic functions are used for the Expanded function set of the Series Six Plus PLC. Press F1 to select the Floating Point Addition function. Shown only in columns 1 to 4 of the first line of a rung. Press F2 to select the Floating Point Subtraction function. Shown only in columns 1 to 4 of the first line of a rung. Press F3 to select the Floating Point Multiplication function. Shown only in columns 1 to 4 of the first line of a rung. Press F4 to select the Floating Point Division function. Shown only in columns 1 to 4 of the first line of a rung. Press F5 to select the Floating Point Compare function. Shown only in columns 1 to 5 of the first line of a rung. Press F6 to select the Integer to Floating Point Conversion function. Shown only in columns 1 to 6 of the first line of a rung. Press F7 to select the Floating Point to Integer conversion function. Shown only in columns 1 to 6 of the first line of a rung. Press F8 to return to the Expanded Arithmetic menu.
F7	Expanded Compare	Check the equality of two values. Valid ranges include inputs, outputs. auxiliary I/O, Expanded I/O, registers to 16K, and constants.
F8	Edit Menu	Return to the basic Edit function key assignments.

Table Move (F3)

Table Move functions transfer data in memory. The specified data is copied from one location to another. It exists unchanged in the original location, and writes over any data at the destination. Press F3 to display the following function key assignments:

Function Key	Function	Description
F1	Source to Table	Copy data into a table, 16 bits at a time. Shown only in columns 1 to 7 of the first line of a rung.
F2	Table to Destination	Copy data from a table to a specified location, 16 bits at a time. Shown only in columns 1 to 7 of the first line of a rung.
F3	Table Move	Copy the contents of one table into another table location. Shown only in columns 1 to 6 of the first line of a rung.
F4	Extended Table Move	Access the registers above 1024. Shown only in columns 1 to 5 of the first line of a rung, if the Scratch Pad is configured for Advanced or Expanded functions. Reference content will be shown for discrete references (I,O, AI, AO, EI and EO) but not for registers.
F7	Coil	Move the cursor to the tenth (coil) column, entering horizontal shunts as necessary. Displays one-shot and relay coil labels.
F8	Edit Menu	Return to the basic Edit function key assignments.

List Functions (F4)

List functions are used to transfer data in memory. The specified data is copied from one location to another. It exists unchanged in the original location, and writes over any data already stored in the destination. Press F4 to display the following function key assignments:

Function Key	Function	Description
F1	Add to Top	Copy a 16-bit value from a specified source to the first location in a list. Shown only in columns 1 to 6 of the first line of a rung.
F2	Remove from Bottom	Copy a 16-bit value from the last location in a list to a specified destination. Shown only in columns 1 to 6 of the first line of a rung.
F3	Remove from Top	Copy a 16-bit value from the first location in a list to a specified destination. Shown only in columns 1 to 6 of the first line of a rung.
F5	Sort Ascending	Sort the values in a list in ascending order, writing over the original data in the list. Shown only in columns 1 to 6 of the first line of a rung.
F7	Coil	Move the cursor to the tenth (coil) column, entering horizontal shunts as necessary. Displays one-shot and relay coil labels.
F8	Edit Menu	Return to the basic Edit function key assignments.

Matrix Functions (F5)

Press F5 to display the following function key assignments:

Function Key	Function	Description
F1	Logical AND	Compare each bit in matrix A with the corresponding bit in matrix B. When both are 1, it places a 1 in the corresponding bit in matrix C. Shown in columns 1 to 6 of the first line of a rung.
F2	Logical Inclusive OR	Compare each bit in matrix A with the corresponding bit in matrix B. When either is a 1, it places a 1 in the corresponding bit in matrix C. Shown in columns 1 to 6 of the first line of a rung.
F3	Logical Exclusive OR	Compare each bit in matrix A with the corresponding bit in matrix B. When they are different, it places a 1 in the corresponding bit in matrix C. Shown in columns 1 to 6 of the first line of a rung.
F4	Logical Invert	Set each bit in matrix B to the opposite of the corresponding bit in matrix A. Shown in columns 1 to 7 of the first line of a rung.
F5	Matrix Compare	Compare the state of each bit in matrix A with the corresponding bit in matrix B. When they are not the same, the result is compared with the corresponding bit mask in matrix C. Shown in columns 1 to 4 of the first line of a rung.
F6	Bit Matrix	The Bit Matrix functions operate on individual bits within a matrix: Press F1 to select the Matrix Bit Set function. This function senses or sets a bit in a matrix. Shown in columns 1 to 6 of the first line of a rung. Press F2 to select a Matrix Bit Clear function. This function senses or clears a bit in a matrix. Shown in columns 1 to 6 of the first line of a rung. Press F3 to select a Matrix Bit Shift Left function. This function shifts all bits in a matrix a specified number of locations to the left. Shown in columns 1 to 6 of the first line of a rung. Press F4 to select a Matrix Bit Shift Right function. This function shifts all bits in a matrix a specified number of locations to the right. Shown in columns 1 to 6 of the first line of a rung. Press F5 to return to the Matrix functions.
F7	Coil	Move the cursor to the tenth (coil) column, entering horizontal shunts as necessary. Displays one-shot and relay coil labels.
F8	Edit Menu	Return to the basic Edit function key assignments.

Control Functions (F6)

Control Functions control or alter the sequence of logic operations. Press F6 to display the following function key assignments:

Function Key	Function	Description
F1	Do Subroutine	Cause a subroutine to be executed. This function can only be placed in the main logic program. Shown in columns 1 to 7 of the first line of a rung.
F2	Return	Select the Return function to end a subroutine. This function can only be programmed in a subroutine program. Shown only in the first column.
F3	Suspend I/O	Cause the CPU to suspend its normal I/O servicing. Not shown in the ninth column. Shown in columns 1 to 8 of the first line of a rung.
F4	Do I/O	Cause the I/O to be serviced immediately, during the scan of the ladder diagram program. Shown in columns 1 to 7 of the first line of a rung.
F5	Status	Open or close the communications window to the DPU and CCM. This function can also be used to control the number of parity check retries. Shown in columns 1 to 7 of any line of a rung.
F6	Configuration	The Configuration functions describe the characteristics of the system: Press F1 to select the CPU Configuration function. This function will store the CPU information entered on the CPU Configuration display, and data. It must be entered in the program to display the CPU Configuration screen. Shown only in column 1 of rung 1 of a program. Press F2 to select the Series 90™-70 I/O Rack Configuration function. This function stores the configuration of the I/O racks entered on the Series 90-70 I/O Rack Configuration screen.
F7	Window	For the Expanded level CPU, selects the Window function. This function is a DPU communications request to support the Expanded I/O system. Not shown in column 10.

SECTION 4
Searching for a Program Element

To search for an element of a program being displayed on the screen, select Search Function (F2) from the Edit or Display Program function. The following new function key assignments are displayed.

Function Key	Function	Description
F1	Begin Search	Begin searching for the element defined in the work area.
F2	Advance	Toggle the search direction between forward and reverse.
F3	Reference Explicit	Toggle the search between explicit (REF EXP) and implicit (REF IMP) references.
F4	From Top	Toggle the starting point for the search between the top of the program and the current cursor location.
F5	Relay Elements	Display the Relay Element Search function keys.
F6	Basic Functions	Display additional Search keys: Shift/Move (F3) Basic Arithmetic (F4) Special Functions (F5)
F7	Advanced Functions	Display additional Search keys: Data Move (F1) Expanded Arithmetic (F2) Search Table (F3) List Functions (F4) Matrix Functions (F5) Control Functions (F6)

1. To search for a reference only, regardless of how that reference is used, type the entry into the reference line of the work area. Continue at step 6 below.
2. To search for a nickname, type it into the text line of the work area. To search for a specific use of a nickname, then select the type of element using the function keys as shown in the listing that follows in this section. Continue at step 4 below.

   An Asterisk (*) can be used to search for similar nicknames. It must be used at the end of the nickname, to replace the variable characters at the end of the nickname. For example, entering PB* would search for PB1, PB2, or PB3. This is known as a ''wildcard'' nickname search.
3. After entering a reference or nickname if desired, press the appropriate search function key(s), as explained on the following pages. This will enter the name of the function into the work area.
4. After you identify the element to be searched for, select the method of searching:

   A. Press F2 (Advance) to toggle the search direction between forward and backward.

B. Press F3 (Explicit Reference) to toggle between:

(1) Explicit References only - to search only for references actually entered in the ladder diagram and displayed on the screen.

(2) Explicit and Implied - to search for both explicit and implied references, which are not displayed, but which are included in tables, matrixes, and similar functions.

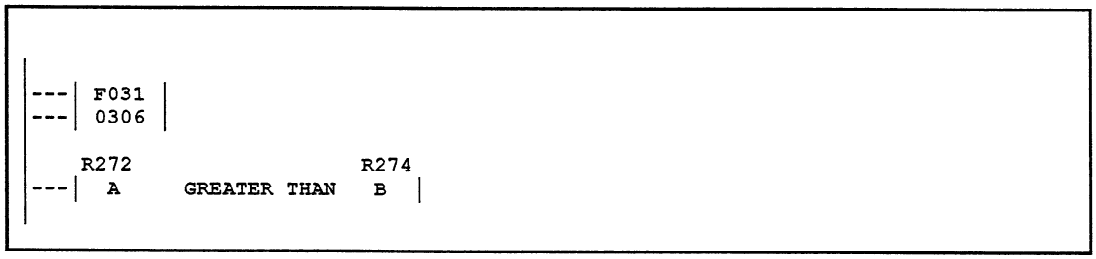
C. Press F4 (From Top) to toggle the starting place for the search between the top of the ladder diagram and the current rung.

5. Press F1 (Begin Search) to begin searching. When the element is located, that element will be displayed at the top of the screen.
6. To repeat the search, adjust the parameters as needed, and press F1 again.
7. To exit from the Search function, press F8 (Program Menu). The system returns to either the Edit Program function.

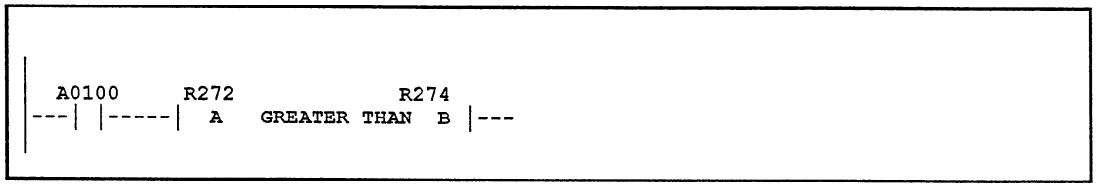
Searching for a "Bad Opcode"

The Bad Opcode (F7) key can be used to locate incorrect program mnemonics. For example, certain commands in programs created with the Series Six™ Program Development Terminal (PDT) are incompatible with Logicmaster 6 software. This is indicated by the message "Unrecognized Opcode" when a program is loaded from the CPU. The Bad Opcode Search function finds only incorrect mnemonics; it does not find incorrect logic.

The following example is an incompatible opcode from a program created on the PDT.



For this rung, you would delete the illegal lines, open a space before the A Greater Than B mnemonic using the Open Space and Move Right Edit functions, and then edit in a compatible reference. The corrected rung might look like this:



SECTION 5
Ladder Diagram File Editing

The file editing capabilities of the Logicmaster 6 software can be used to store part of a ladder logic program in a separate file, or to merge part or all of one program with another.

Copying Rungs to a Side File

To copy rungs from a program into a Side file:

1. With the program displayed in Edit mode, count the exact number of rungs to place in the Side file.
2. Place the cursor at the first rung to be copied.
3. Use the Select key to move the work area cursor to the upper line and enter a name for the file. The system will automatically add the extension .SDE to this file name when the file is created.
4. Use the Select key to move the work area cursor to the bottom line, and enter the number of rungs to be copied to the file. Then, press ALT-W. The system creates a new file, consisting of the specified rungs.

Adding an .SDE or .LAD File to a Program

The entire content of an .SDE or .LAD file can be added to the ladder logic program if no file name is active. The time needed to merge two ladder logic files depends on the size and complexity of both files. Simple programs can be merged in just a few seconds. For long and complex programs, file merging may take several minutes.

Time needed for file merging also depends on the place in the first program where the new logic is added (since each existing rung must be moved back to make room for the new rungs). For fastest file merging, add the new logic as close as possible to the end of the first program.

To combine files:

1. Display the program that will receive the added rungs. Go to Edit mode. Place the cursor at the last rung before the place you want to add the rungs. When you merge the files, the added logic will appear immediately after the highlighted rung.
2. Use the Select key to move the work area cursor to the top line, and enter the name of the .SDE or .LAD file. If the file is located on a drive other than the one which is currently active, begin with the drive ID followed by a colon. For a .LAD file, you must enter the extension .LAD after the file name. For example:

```
b:program1.lad
```

3. Press ALT-G. The start of the other ladder logic appears at the cursor position.

Checking a Merged Program

Using the file merge functions can create an incorrect program. Be sure to check the program you create for correct format. If you try to store an incorrect program (either to the CPU or to disk), the system will issue an error message, and the program will not be stored.

Some things to check are:

- If the program has a CPU Configuration function, it must be located at rung 1.
- If you merge two .LAD files and each has its own CPU Configuration function, one must be deleted. The CPU Configuration function contains the information from the CPU Configuration screen (see chapter 10).
- If you delete one CPU Configuration function from the program, be sure that the one that remains is correct.
- The maximum number of subroutines allowed is 16.
- After each subroutine, there is a Return function.
- Between a main program and a subroutine there is one End of Sweep function.
- After the end of the entire program, there must be two End of Sweep functions together.

Chapter 6 Annotation

Annotation is an important feature of the Logicmaster 6 software. It can be used to add explanatory text to a ladder logic program. The annotation can be names and nicknames for individual references, or longer blocks of text used for rung explanations and coil labels within the program.

Nicknames: A program reference can be assigned a "nickname" of up to 7 ASCII-keyboard characters. Each nickname must be unique within a program. Examples of nicknames are LS035, PB11A@, SOL129C. Every discrete reference and all registers from R0001 to R16384 can be assigned a nickname. The total number of nicknames permitted in a program is either 2048 or 5120, depending on the selection made for "windowing" on the Machine Setup menu. The Machine Setup menu is reached from the Expanded Functions menu. For information, refer to chapter 10, *Expanded Functions Menu.*

Names: Every program reference can also be assigned a "name" of up to 21 ASCII-keyboard characters. Names are formatted as three lines of 7 characters each. Reference names do not have to be unique within a program. Examples of names are: HIGH LEVEL ALARM, ERROR CONDITN DETECTD. Every discrete reference and all registers (from R0001 to R16284) can be assigned a name. The total number of names and nicknames permitted in a program is limited only by the amount of space available on the disk.

Rung Explanations: For each rung, an explanation can be entered. The explanation might describe the logic function, or provide diagnostic information for maintenance personnel. A rung explanation can have up to 4000 ASCII-keyboard characters, entered up to 255 characters per line. The display will show only the first 74 characters, with a diamond symbol indicating the presence of additional characters.

Coil Labels: Each coil (O and AO references, and external and internal output channels 0-F) can be assigned a label to represent its function, or its output device. Coil labels can contain up to 4000 ASCII-keyboard characters, entered up to 255 characters per line. The display will show only the first 46 characters, with a diamond symbol indicating the presence of additional characters. Coil labels appear above the coil on the screen, and in 80-column printouts. They appear to the right of the coil on 132-column printouts.

Using Annotation in a Program

To use annotation in a program, a file name must be specified. Enter a program name of up to 8 characters in the text line of the work area before selecting Display Program, Edit Program, or Print from the Supervisor menu.

Loading the Program

Follow the steps below if the program to be annotated is not already displayed.

1. If the program is stored on disk, type in the name of the program and press CTRL-E (or the Enter key). Then, press ALT-L to load the program from the Supervisor menu. Continue at "Creating an Annotation File" below.
2. If the program is the one now in the CPU, use the Load/Store/Verify functions to load it into memory.
   A. From the Supervisor menu, select Load/Store/Verify (F6).
   B. In the Load/Store/Verify menu, press F1 (Load function).
   C. Enter the drive ID and program name. Then, press CTRL-E (or the Enter key) and continue below.

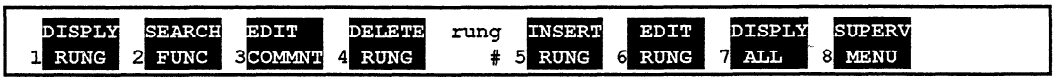
Creating an Annotation File

The program name must be entered from the Supervisor menu to use annotation. If you have not entered the file name, follow the steps below:

1. From the Supervisor menu, type in the file name for the program.
2. To create the file without entering Edit Program mode, press the Enter key. The file name will be displayed below the menu selections.
3. Or press the Edit Program (F2) function key. When the backup prompt appears, enter **N** if no annotation file with the same name already exists, or **Y** to save a previous version of the file before editing the annotation.

Accessing the Annotation Editing Functions

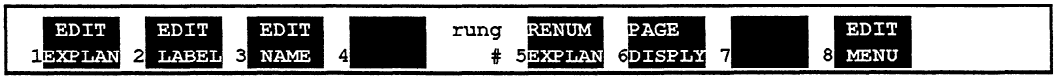
Annotation editing functions are accessed through the Edit menu.



Two of these Edit Mode function keys are used with annotation:

Function Key	Function	Description
F7	Display All	Toggle the format of the program display between reference (nickname-only) display and annotated (reference, nickname, name) display. The .NAM file for the program must be present in the default drive for proper annotation display.
F3	Edit Comment	Access the annotation entry and editing functions.

Pressing the Edit Comment (F3) key displays the following function key assignments:



Function Key	Function	Description
F1	Edit Explanation	Enter or edit rung explanations.
F2	Edit Label	Enter or edit coil labels.
F3	Edit Name	Enter or edit names and nicknames.
F5	Renumber Explanations	Renumber rung explanations if rungs have been added or deleted in the program at a time when the file containing the annotation was not present.
F6	Page Display	Toggle between Page Display and Window Display mode.

Nicknames

A program reference can be assigned a "nickname" of up to 7 ASCII-keyboard characters. Each nickname must be unique within a program. Every discrete reference and all registers from R0001 to R16384 can be assigned a nickname.

The total number of nicknames permitted in a program is either 2048 or 5120, depending upon the selection made for "windowing" on the Machine Setup menu. The Machine Setup menu is reached from the Expanded Functions menu. For more information, refer to chapter 10, *Expanded Functions Menu.*

Nicknames and references can be changed on-line. However, a nickname cannot be added to a reference on-line.

Entering Nicknames in a Program

Nicknames can be entered when a rung is created, or entered separately. To enter a nickname while creating a rung:

1. Be sure that the program name has been specified from the Supervisor menu.
2. While entering a program element, the reference is entered on the center line of the work area. To assign a nickname for that reference, enter up to 7 ASCII-keyboard characters on the top line of the work area. For example: LS035, PB11A@, SOL129C.
3. Press the Shift key and the Enter key at the same time, or press an instruction function key. The system checks whether the reference has been used before in the program, and whether the reference has already been assigned a nickname. If not, that nickname will be assigned to the reference throughout the program. The nickname will be stored on the program disk when the rung is accepted.

If the nickname has already been used, a message is displayed and the nickname is not stored. If the reference has a prior nickname, a message is displayed requesting confirmation to delete the prior nickname. If confirmed, the new nickname will replace the former one throughout the program. It will be stored on the program diskette.

Deleting a Nickname

To delete a nickname, enter the reserved name "none" as the nickname in the top line of the work area. Enter the reference from which the nickname is to be deleted in the center line of the work area. Press the Shift-Enter keys at the same time, or select an instruction function key. A message will be displayed, confirming the nickname deletion.

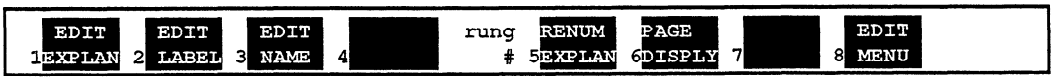
Entering Annotation in Window Mode

In addition to entering nicknames when creating a rung, annotation can be entered in either Window mode or Page mode. In Window mode, a window is created in the ladder logic. In this way, the annotation can be entered while the logic is displayed on the screen.

When entering rung explanations or coil labels, which can contain many lines of text, a beep occurs near the end of a display line. This is similar to the end-of-line bell on a typewriter.

To enter annotation in Window mode:

1. From the Edit menu, press F3 (Edit Comment).



   The F3 key is available only when Edit Comment is selected from Display All mode and the cursor is at a rung element with a register or discrete I/O reference. The F2 key is available only when the cursor is on a rung with a coil.

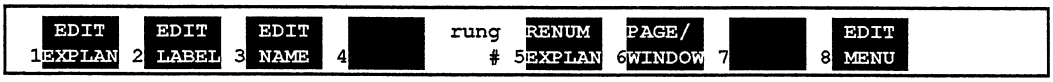
2. If the F6 key is labeled Window Display, press F6 to enter Window mode.
3. With the cursor at the rung where the annotation will appear, press F1 to enter a rung explanation, F2 to enter a coil label, or F3 to enter a name. A reverse-video window appears above the rung. The window is seven lines deep and 74 characters wide for a rung explanation; for a coil label, the window is seven lines deep and 46 characters wide.
4. Enter text into the window. The edit window will scroll upward when the bottom is reached, allowing entry of up to 4000 characters.
5. After entering the text, press CTRL-A (or the Accept key) to save it. Or, press F9 (or the Abort key) to delete it. Saved text will be stored in the program's annotation file.

Entering Annotation in Page Mode

Annotation can also be entered in Page mode. In Page mode, the entire screen is used for annotation. This enables you to enter annotation prior to actually creating the ladder logic. The annotation can then be saved in a file, for future use.

To enter annotation in Page mode:

1. In the Edit menu, press the Edit Comment (F3) key.



2. If function key F6 currently is labeled Page Display, press F6. The ladder logic disappears from the screen.
3. Use the function key (F1, F2, or F3) which corresponds to the type of annotation you wish to enter:
   A. For a rung explanation, press CTRL-S (or the Select key) to move the work area cursor to the bottom line. Enter the rung number above which the rung explanation is to appear. The Increment and Decrement keys (these are shifted key functions) can be used to enter rung explanations in sequential order. Then, press F1. A reverse-video window appears for entering the text of the rung explanation; the window is 21 lines deep and 74 characters wide.
   B. For a coil label, enter either the coil nickname in the top line of the work area, or the coil reference in the center line of the work area. The Increment and Decrement keys can be used to enter coil labels in sequential order. Then, press F2. A reverse-video window appears for entering the text of the coil label; the window is 21 lines deep and 46 characters wide.
   C. For a name or nickname for a reference, enter either the nickname in the top line of the work area, or the reference in the center line of the work area. The Increment and Decrement keys can be used to enter references in sequential order. Then, press F3. The following display appears. Here, the illustration assumes that I0001 was entered in the second line of the work area.

```
NAME FIELD #1 ----------------
NAME FIELD #2 ----------------
NAME FIELD #3 ----------------
NICKNAME FIELD ---------------
REFERENCE FIELD ADDRESS ------
```

4. Enter text into the edit window. The window will scroll upward when the bottom is reached, allowing up to 4000 characters to be entered.
5. After the text is entered, press CTRL-A (or the Accept key) to save it. Or, press F9 (or the Abort key) to delete it. The text will be stored in the program's annotation file.

Editing Annotation Text

In an edit window, the end of the explanation or label is marked EOB (End of Block). If there is no text in the window, the EOB appears in the upper left corner of the window. Enter new text by typing it in. When the text is complete, press CTRL-A (or the Accept key) to save it in an annotation file with the same name as the program file name.

The following function key assignments are available for editing annotation text.

Function Key	Function	Description
F1	Upper Functions	Toggles the active functions of the other keys.
F2	UND C/DEL C	F2 (DEL C) deletes the character at the cursor position, and moves the following text one character to the left.
		With the upper functions active, F2 (UND C) inserts the most recently-deleted character immediately before the cursor position, and moves the cursor one character left, to the new character's position. This can be done repeatedly, to insert the most-recently deleted character at several positions in the text.
F3	UND W/DEL W	F3 (DEL W) deletes all the characters from the cursor position to the next word or the next carriage return.
		With the upper functions active, F3 (UND W) inserts the most recently-deleted character string (word segment) immediately before the cursor position.
F4	UND L/DEL L	F4 (DEL L) deletes all characters from the cursor position to the end of the line, including the carriage return.
		F4 (UND L) inserts the most recently deleted line of text immediately before the cursor position.
F5	DELEOL/EOL	F5 (EOL) moves the cursor to the end of the line.
		F5 (DELEOL) deletes all characters from the cursor to the end of the line, but not the carriage return at the end of the line.
F6	Reset/Select	F6 (Select) marks a location in the window from which text will be removed using the Cut (F7) key. After pressing the Select key, move the cursor to indicate the area from which text will be removed.
		F6 (Reset) resets (or deselects) the selection made with the Select key.

Function Key	Function	Description
F7	Paste/Cut	F7 (Cut) removes the characters selected with the Select (F6) key. These characters are placed in a buffer, to be placed elsewhere in the text window using the Paste key.
		F7 (Paste) inserts the contents of the buffer immediately before the cursor location. This can be done more than once, to make multiple copies of the same text. The text can also be restored to its original location using the Paste key.
F8	Top/Bottom	F8 (Bottom) moves the cursor to the last character of the text block.
		F8 (Top) moves the cursor to the first character of the text block.

Text editing must be ended either by pressing:

- CTRL-A (or the Accept key).
- F9 (or the Abort key).
- The Escape key (or the Supervisor key).

Creating Additional Text Files

The maximum number of characters that can be included in a rung explanation or coil label is 4000. However, longer text can be incorporated in a program printout by creating an annotation text file. Such text files can only be printed, not displayed on the screen.

To include a text file in a printout, follow the steps below:

1. Create the file, using one of the many DOS-compatible word-processing software packages.
2. Store the text file on a drive that is accessible to the system during the printout.
3. Create the coil label or rung explanation. At the place in the label or explanation where the text file is to be included in the printout, move the cursor to the beginning of a new line, and enter **I**, followed by the drive designation (if necessary) and the name of the file. There must be no other text on the same line. For example:

```
I B:LABEL:MEM
```

Renumbering Rung Explanations

Rung explanations are stored in a different file from the ladder logic for the program. If rungs are added or deleted in the program, the system will automatically re-structure the text in the annotation file so that the explanations will appear in the correct locations in the program.

However, if the annotation file is not present when the program is changed (i.e., the diskette containing the annotation file is not in the drive) or during file merge operations (i.e., including rungs from an .SDE file), the system will not be able to perform this automatic re-structuring on the annotation file. In that case, the annotation must be changed to agree with the new program content. If rungs were added or deleted when the annotation file was not available, follow these steps:

1. **Window Mode:** If rungs were added, move the cursor to the first new rung of logic. If rungs were deleted, move the cursor to the first rung after the deletion.

Page Mode: If rungs were added, type the rung number of the first new rung of logic, on the center line of the work area. If rungs were deleted, type the rung number of the first rung after the deletion, on the center line of the work area.

2. In the bottom line of the work area, enter a plus (+) sign to indicate added rungs, or a minus (-) sign to indicate deleted rungs. Follow that with the number of rungs added or deleted.

NOTE

The use of incorrect values may result in a loss of more explanations than intended.

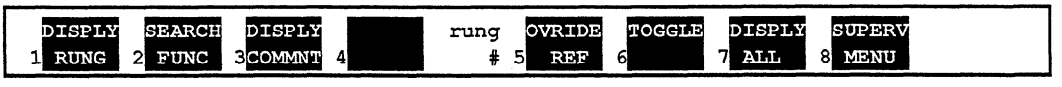
3. Select the Renumber Explanations (F5) key from the Comment Editing display. The screen displays the following message:

```
PRESS  ALT-X (CONFIRM)  TO MOVE REST OF EXPLANATIONS - SOME MAY BE LOST
```

4. Press ALT-X (or the Confirm key) to execute the re-numbering. The rung explanations will be re-numbered in the direction, and by the quantity entered.

Viewing Annotation in Display Program Mode

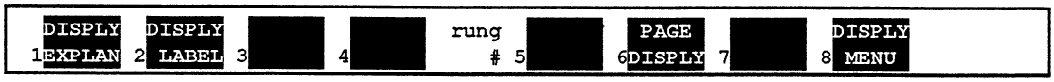
Annotation Display functions are accessed through the Display Program menu.



The Display Program menu provides two function keys for annotation display:

Function Key	Function	Description
F7	Display All	Toggle the format of the program display between reference (nickname-only) display and annotated (reference, name, nickname) display. The .NAM file for the program must be present in the default drive for proper annotation display.
F3	Display Comment	Access the Annotation Display functions.

Pressing the Display Comment (F3) key displays the function key assignments:



Function Key	Function	Description
F1	Display Explanations	View rung explanations.
		If the Window Display (F6) function has been selected and Window mode is active, pressing F1 will display the rung explanation of the rung where the cursor is located. The rung explanation appears in a reverse-video window that is seven lines deep and 74 characters wide. If there is additional text, use the cursor to scroll the text in the window.
		If the Page Display (F6) function has been selected and Page mode is active, pressing F1 will display the rung explanation of the rung number entered in the bottom line of the work area. The rung explanation appears in a reverse-video window that is 21 lines deep and 74 characters wide. If there is additional text, move the cursor to scroll the text in the window.
F2	Display Label	Display coil labels. F2 is only available when the cursor is on a rung within a coil. It is always available in Page mode.
		If the Window Display (F6) function has been selected and Window mode is active, pressing F2 will display the coil label of the coil on the rung where the cursor is located. The coil label appears in a reverse-video window that is seven lines deep and 46 characters wide. If there is additional text, use the cursor to scroll the text in the window.
		If the Page Display (F6) function has been selected and Page mode is active, pressing F2 will display the coil label of the coil reference entered in the center line of the work area, or the nickname of the coil reference entered on the top line of the work area. The coil label appears in a reverse-video window that is 21 lines deep and 46 characters wide. If there is additional text, use the cursor to scroll the text in the window.
F6	Page Display	Toggle between Page Display and Window Display mode.

Printing Annotation

Annotation can be printed with the ladder program, or separately. Refer to chapter 8, *Print Functions*, for more information.

To set up the printer:

1. If the printer has already been set up and the printout parameters specified, go directly to step 2. Otherwise, set up the printer (this is only required when using a printer for the first time).
   A. If it is a parallel printer, attach it to port 3. Turn the printer on, and place it in On-Line mode. Continue at step 1C.
   B. If it is a serial printer, set the DIP switches and jumpers on the printer, as instructed in the manual for the printer. Attach the printer to serial port 1. From the Supervisor menu, select Utility Functions (F8). Enter parameters for port A which correspond to those set up for the printer. Press the Set Up Port (F1) key to implement the printer parameters at the port.
   C. Return to the Supervisor menu. Select Print functions (F5). If no file name is active, enter the file name of the annotation to be printed, on the top line of the work area.
   D. Define the printer parameters. From the Print menu, press F5 (Define Printer). The default printer parameters can be changed, as described in chapter 8, *Print Functions*.
2. From the Print menu, press F6 (Define Output) to determine the content of the printout. Here, you can select the program and annotation feature to be printed.
3. To print the annotation in foreground mode, select Print Out (F1) from the Print menu. This command can either send the output to a printer directly, or store it as a print file for printing later in background mode. Specify the port to receive the printout or file. If the file is being saved for later printing, give it a file name. Press CTRL-E (or the Enter key) to start printing.
4. To print the annotation in background mode, select Print Program (F2) from the Print menu. Specify the printer port and the source drive for the file. Enter the program name, and press CTRL-E (or the Enter key) to start printing.

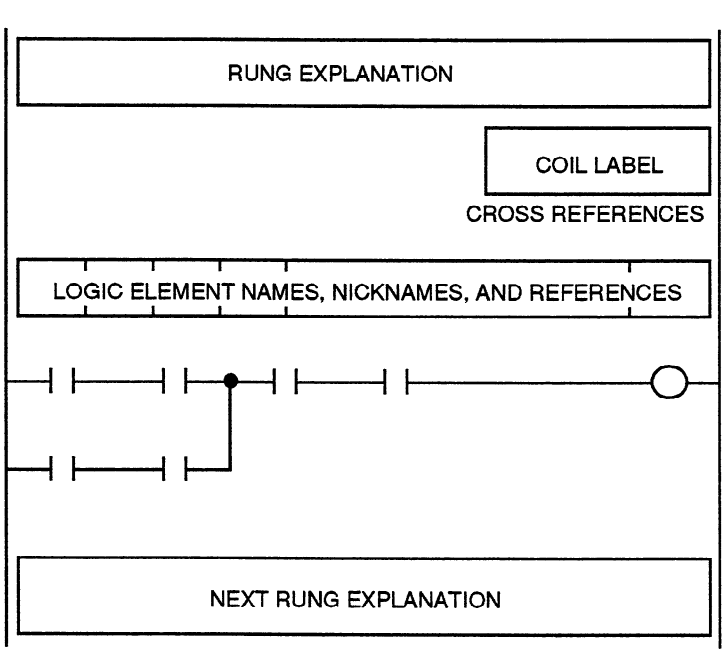


Figure 6-1. 80-Column Format of Printout with Annotation

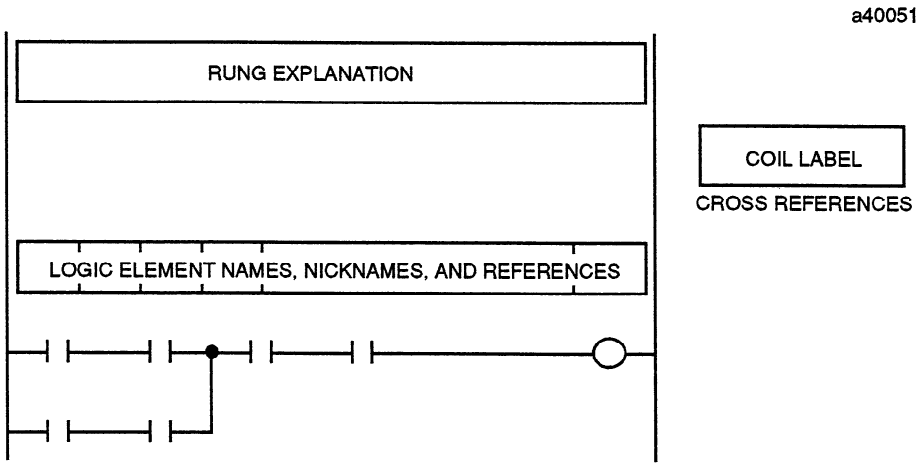


Figure 6-2. 132-Column Format of Printout with Annotation

Changing the Title on the Printout

When the printer output is defined, a title and subtitle for the printout are created. This title and subtitle can be changed before any rung explanation in the program. Each time a new title and subtitle are encountered during printout of the program, the printer begins a new page. The new title and subtitle appear at the top of the page, before the rung explanation, and will continue until it is changed again by another rung explanation.

Follow the steps below to change the title and/or subtitle for the printout pages:

1. At the beginning of the rung explanation, enter the character **\T** for a new title or **\S** for a new subtitle.
2. Type up to 60 characters for the title or subtitle, on the same line as the \T or \S.
3. Continue the text of the rung explanation on the next line.

Placing Borders around Comments in Printouts

To make coil labels and rung explanations stand out in program printouts, borders can be printed around them. The system will automatically create a border of the correct size. To place a border around a label or explanation, follow the instructions below:

1. Create the coil label or rung explanation.
2. At the beginning of a line by itself, enter **\B** followed by two ASCII characters to make up an outer and inner border. The two ASCII characters must be the next two characters after the \B. Any spaces in these two positions will become part of the border. For example:

```
\B* (space)
```

This would create a border of asterisks around the outside of the comment. A blank is usually specified as the inner (second) character, to provide space between the text and the border.

The \T, \S, and \B can be entered on any of the first three lines of the rung explanation, in any order. For example:

```
\S      New subtitle
\B*     Border
\T*     New title
```

For coil labels, the \B can be any of the first three lines of the text.

Borders are used only in printouts; they do not appear on the screen.

Chapter 7
Display Reference Tables

The Display Reference Tables function is used to display the status of any group of references. These may be discrete references (I/O, Auxiliary I/O, or Expanded I/O) or registers.

The Display Reference Tables function can be selected from the Supervisor menu or another screen on which it is active. In Off-Line mode, the Display Reference Tables function displays the tables for the program currently in system memory. In On-Line mode, tables values from the program in the CPU are displayed.

CAUTION

When displaying reference tables, if you go from Off-Line to On-Line mode and back to Off-Line, any table you have displayed in the meantime will be updated in the program, and on disk if there is an active file name. If there is a program on the disk with the active file name, the tables for that program will be updated. If the active file name is not the same as the name of the program in the CPU, incorrect values may be written to the program on the disk.

Use reasonable care changing reference table values in On-Line mode.

To display the reference table values for a program:

1. If the program is not currently in system memory, you can load it using the Load/Store/Verify functions, or the quick-load feature described in this step:
   A. Go to the Supervisor menu.
   B. If the file name is not the active file name, type it in. A drive ID followed by a colon can be used before the program name if the program is not on the default disk. Press CTRL-E (or the Enter key).
   C. When the file name is active, press ALT-L to load the file. Go to step 2.
2. In the Supervisor menu, select the reference table to be displayed:
   A. To view the table that was last displayed, or the lowest-numbered Input table if no table was displayed before, make no entry in the work area.
   B. To view a table containing a specific reference, enter the reference or its nickname.
   C. To view the lowest-numbered table of a particular type, enter just the reference type.
   D. To view one of the mixed reference screens, enter its number (1-8).
3. Select Display Reference Tables (F3). The table will appear.
4. To go directly to a different table, type a reference or the nickname for a reference into the work area and press F8 (Display Reference Table).
5. To return to the Supervisor menu, press the Escape key (or the Supervisor key).
6. To return to the ladder diagram display from one of the reference tables displays, press the Escape key (or the Supervisor key) and then press F1 (Display Program). The ladder diagram appears, at its last viewing position.

Fast Update - Serial Versions

Serial versions of the software provide fast update of a selected reference in Display Reference Tables mode. The Fast Update function updates the selected reference very rapidly, while other references are updated more slowly.

To use the Fast Update function, the system must be in On-Line mode with CPU communications active, and the program in memory must be equal to the program in the CPU. Move the cursor to the location of the reference to be updated, and press ALT-U. The screen displays the message "Fast Update Mode."

Fast Update will continue until any other key is pressed, the keyswitch is turned off, or the program in memory and the program in the CPU are no longer equal.

Discrete References

Discrete display screens show status tables of each of the discrete reference types (inputs, outputs, auxiliary inputs and outputs, or expanded inputs and outputs). The data can be displayed in hexadecimal, decimal, signed decimal, or binary format.

When displayed in binary format, the data in the main input/output status table may be overridden. When the system uses the normal mode of I/O addressing, this status table corresponds to I0001 through I1024. When the system uses the expanded mode of I/O addressing, it corresponds to any physical I/O point installed in channel 0 at relative references 0001 through 1024.

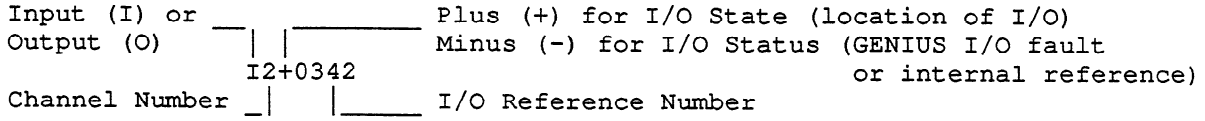
The override state of an I/O point is changed with the Override Reference/Cancel Override function key. Overridden reference points flash on the display. The value of an overridden point can be changed with the Toggle Reference function key.

Normal or Expanded I/O Addressing

Normal mode I/O addressing provides access to 1024 input and 1024 output references in the main chain of I/O. Of these, only 1000 inputs and 1000 outputs are available for installing physical I/O devices. If an Auxiliary I/O module is installed, an additional 1024 input/output references (1000 physical inputs and 1000 outputs) are available.

Expanded mode I/O addressing provides up to 8 channels of I/O on the main chain, and 8 channels on the auxiliary chain. The actual number used depends on the hardware configuration of the system.

Expanded I/O references have the following format:



Displaying a Table of Discrete References

To display one of the I/O reference tables, from the Supervisor menu or from within the Display Reference Tables function, type the reference or its nickname into the work area.

For I/O references, enter the address specification into the work area:

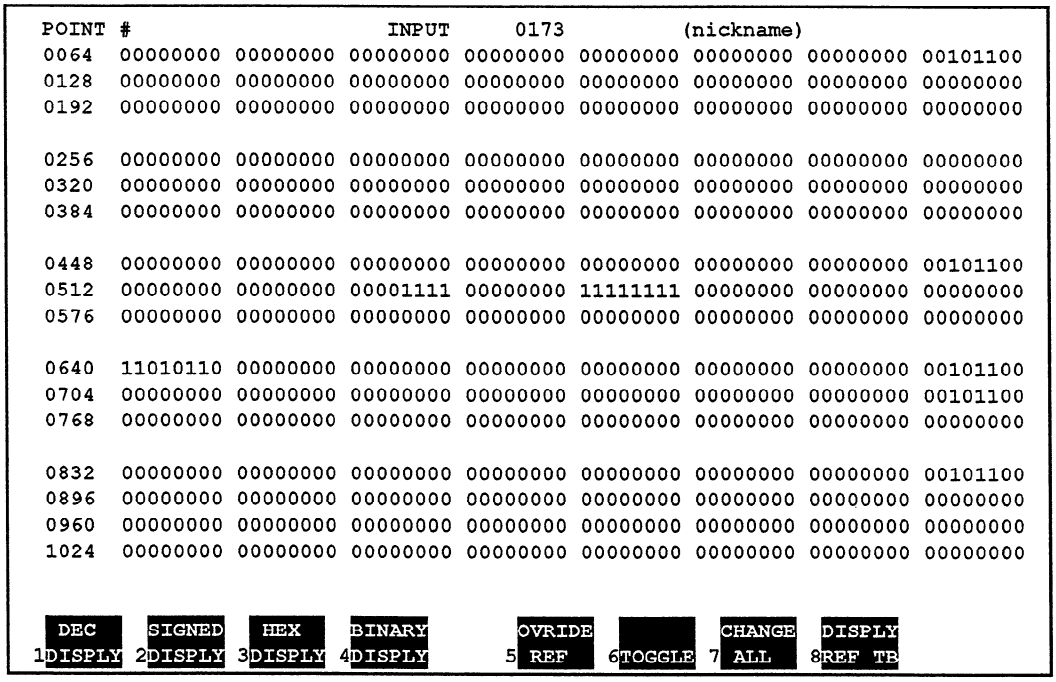
```
I0001
|||__|
|  |  |_____ Point Address (1-1024)
|  |________ I or O (Input or Output)
```

For Auxiliary I/O, enter the address specification:

```
AI0001
|  |__|
|  |  |_____ Point Address (1-1024)
|  |________ AI or AO (Auxiliary Input or Auxiliary Output)
```

On the Discrete Reference Table display, point numbers are indexed in the left column. Discrete values are shown as 0 (off) or 1 (on). Overridden references blink.

For Expanded I/O, line 2 displays the word STATUS, if an internal point is selected, or STATE, for a physical I/O point. The channel number, if any, and the entire reference address are displayed next to the point's nickname.



The following function keys are displayed at the bottom of the Discrete Reference display:

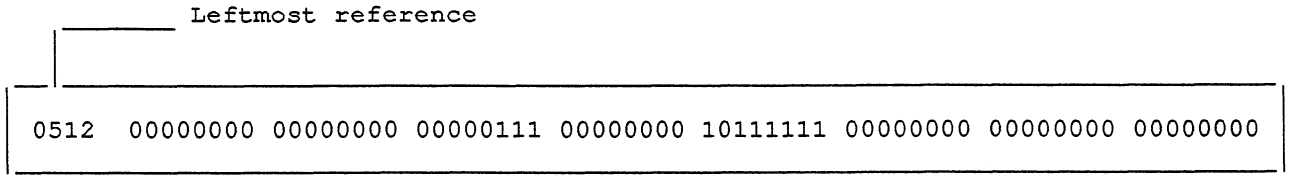
Function Key	Function	Description
F1	Decimal Display	Display values in decimal format.
F2	Signed Display	Display values in signed decimal format.
F3	Hexadecimal Display	Display values in hexadecimal format.
F4	Binary Display	Return the entire screen to binary status display.
F5	Override Reference	Toggle the override of the reference at the cursor position (not available for Expanded I/O; only available for Auxiliary I/O with the Expanded II instruction set).
F6	Toggle	Toggle reference at the cursor position on/off.
F7	Change All	Display all references in the selected format.
F8	Display Reference Table	Display the table containing the reference currently indicated. On a mixed screen, type 1-8.

Press the Escape key (or the Supervisor key) when you want to return to the Supervisor menu from one of the reference tables displays.

To return to the ladder diagram display from one of the reference tables displays, press the Escape key (or the Supervisor key) and then press F1 (Display Program). The ladder diagram appears, at its last viewing position.

Displaying Discrete References in Non-Binary Format

When the screen is in binary format, the reference display shows values in groups of eight bits. This is one line:

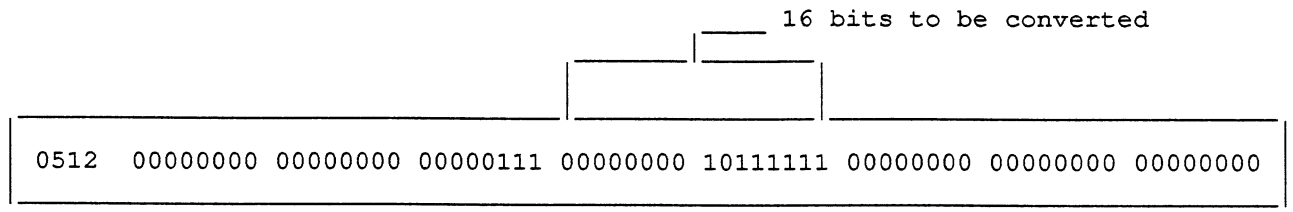


You can change the format of the display to show some or all of the values in decimal, signed decimal, or hexadecimal format. After the display is set up, exiting to the Supervisor menu automatically saves the new format in the reference display format (.RDF) file if there is an active file name. When you return to the reference table display in the future, it will have the new format.

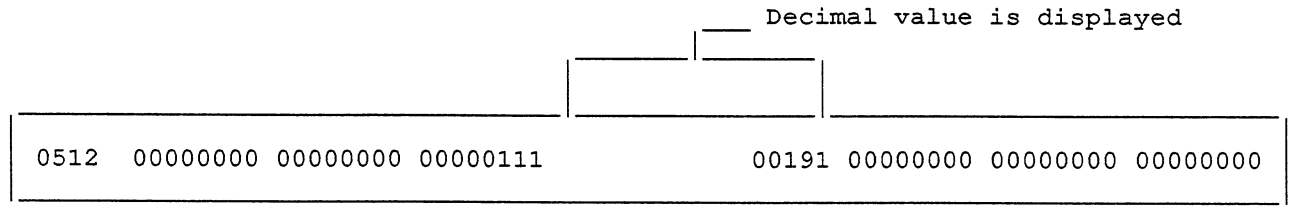
Displaying 16 Bits in Decimal Format

To display the value of 16 bits as a decimal number (00000 to 65535):

1. With the reference display on the screen, move the cursor to the first (lower-numbered) group of eight bits (low byte) to be converted. For example:



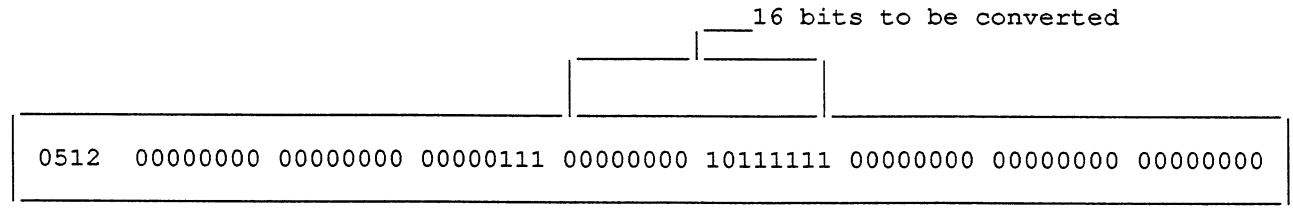
2. Press the Decimal Display (F1) key, or press the DEC/0 key on the 91-key keyboard. The value of the 16 bits is right-justified within the location of the 16 converted bits:



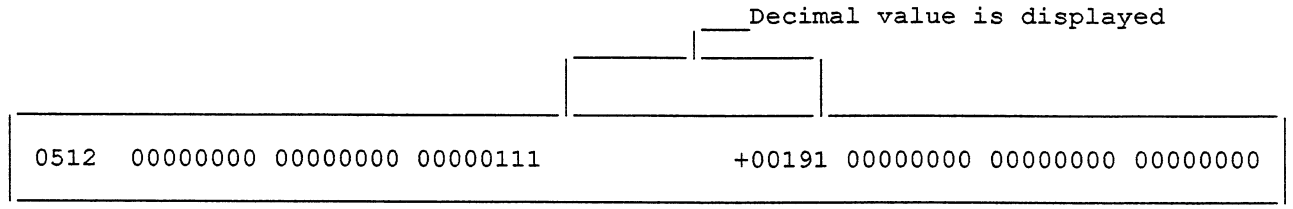
Displaying 16 Bits in Signed Decimal Format

To display the value of 16 bits as a signed decimal number (-32768 to +32767):

1. With the reference display on the screen, move the cursor to the first group of eight bits (low byte) to be converted. For example:



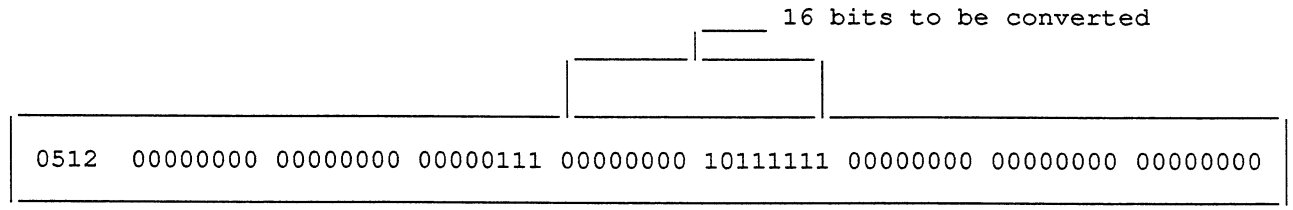
2. Press the Signed Decimal Display (F2) key, or press the ±DEC/R key on the 91-key keyboard. The value of the 16 bits is right-justified within the location of the 16 converted bits. The leftmost binary value is used as the "sign bit". The digit 0 in the leftmost position means the number is positive. The digit 1 in the leftmost position means the number is negative. Because the leftmost of the 16 bits in the example above is a 0, it is converted to a positive number.



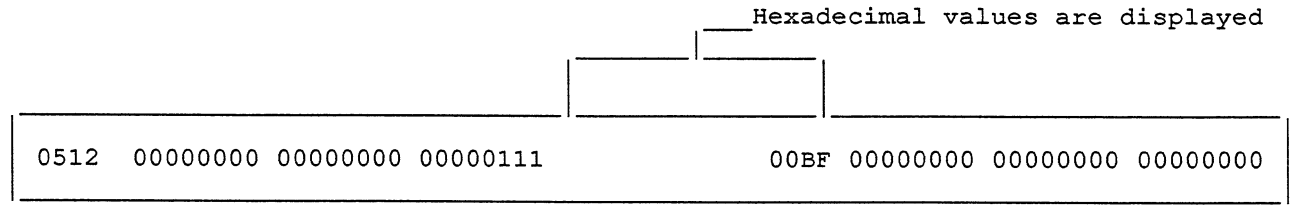
Displaying 16 Bits in Hexadecimal Format

To display the value of 16 bits as four hexadecimal values (0-9, A-F):

1. With the reference display on the screen, move the cursor to the first group of eight bits (low byte) to be converted. For example:



2. Press the Hexadecimal Display (F3) key, or press the HEX/I key on the 91-key keyboard. The value of the 16 bits are converted to four hexadecimal values:



Returning the Display to All Binary Format

After changing the display in one of the ways described above, to return to all binary format press the Binary Display (F4) key.

Converting All References to Another Format

To convert all the references to binary, decimal, signed decimal, or hexadecimal format:

1. With the reference display on the screen, press the Change All (F7) key.
2. Next, press the function key for the display type you want:

Function Key	Function	Description
F1	Decimal Display	Display all values in decimal format.
F2	Signed Display	Display all values in signed decimal format.
F3	Hexadecimal Display	Display all values in hexadecimal format.
F4	Binary Display	Display all values in binary format.

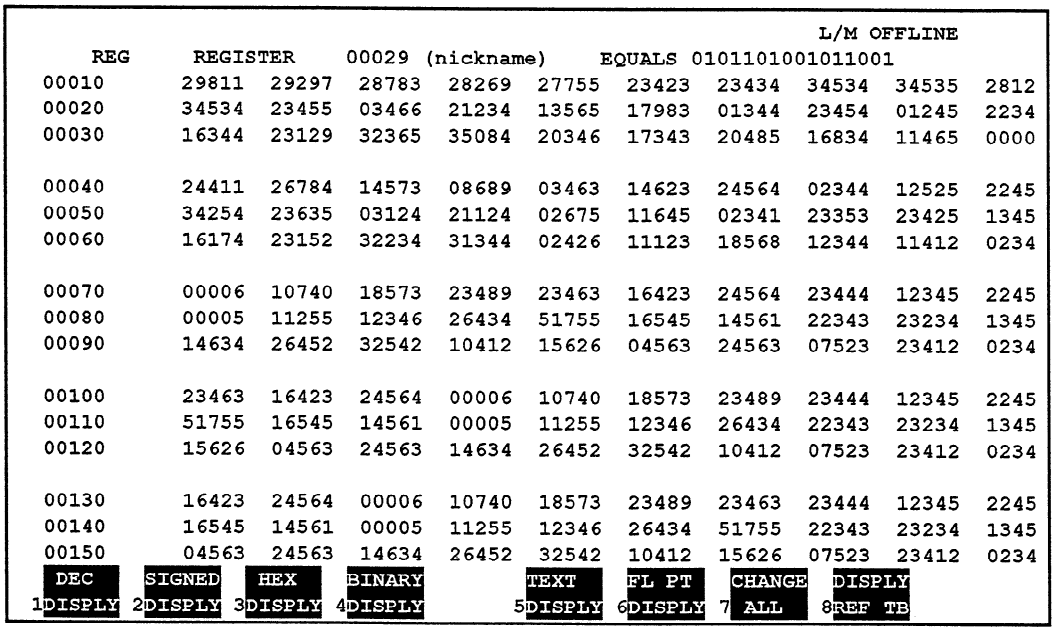
Changing the Base of the Work Area to Floating Point

When displaying I/O tables, you can change the base of the numeric line of the work area to floating point. To change the base to floating point, press the ALT and C/DP keys. Press the Clear key when you want to remove the floating point format.

Register References

To display one of the register tables, type the reference or nickname into the work area. Then, select Display Reference Tables (F3) from the Supervisor menu, or press F8 from another table. The first time this screen is entered, register contents are shown in unsigned decimal format (00000 to 65535).

The top of the Register Reference display shows the register number and nickname indicated by the cursor position (register 00029 in the example above). Also shown is the value's binary equivalent. The left side of the screen lists the register numbers of the left column of values. The lowest register is in the upper right of the display. The highest is at the bottom left.



If the function level selected in the Scratch Pad display is Expanded functions, the following key assignments are displayed:

Function Key	Function	Description
F1	Decimal Display	Display values in decimal format.
F2	Signed Display	Display values in signed decimal format.
F3	Hexadecimal Display	Display values in hexadecimal format.
F4	Double Precision	Display values in double-precision format.
F5	Text Display	Display values as ASCII text, left to right.
F6	Floating Point	Display values as floating point numbers.
F7	Change All	Display all references as selected by F1-F4.
F8	Display Reference Table	Display the table that includes the reference currently entered in the work area.

If the function level selected in the Scratch Pad display is less than Expanded, key assignments are the same as above, except:

Function Key	Function	Description
F5	ASCII Display	Display values as ASCII characters, right to left.
F6	Text Display	Display values as ASCII text, left to right.

Changing the Format of the Register Display

The reference display shows register contents of lines of 10 registers. This is one line:

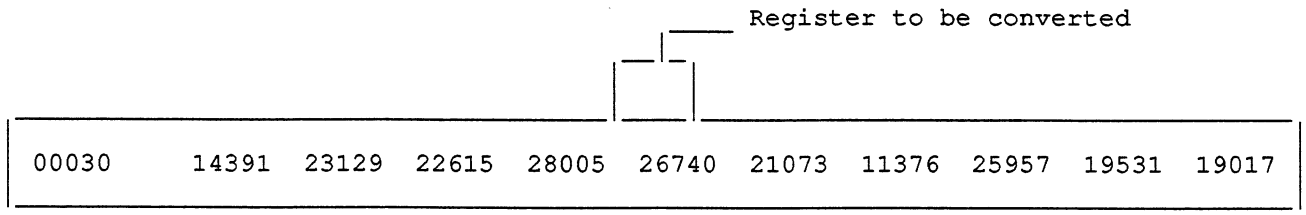
```
00030     14391  23129  22615  28005  26740  21073  11376  25957  19531  19017
```

You can change the format of the display to show some or all of the values in Floating Point, ASCII, or Text format. After the display is set up, exiting to the Supervisor menu automatically saves the new format in the Reference Display Format (.RDF) file (if there is an active file name). When you return to the reference table display in the future, it will have the new format. In addition, the new format will be used when printing out register contents.

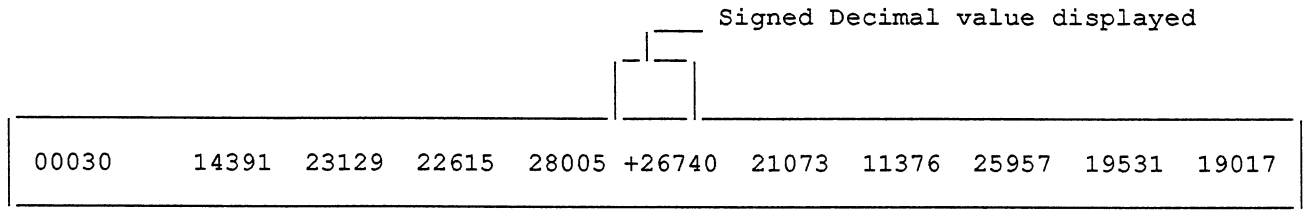
Displaying One Register in Signed Decimal Format

To display the contents of one register as a signed decimal number (-32768 to +32767):

1. With the reference display on the screen, move the cursor to the register to be converted.



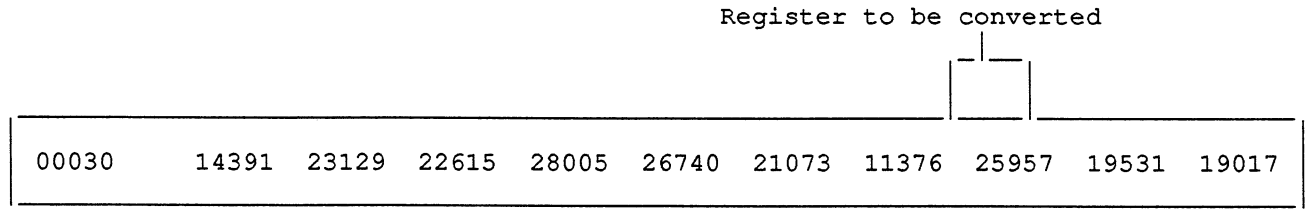
2. Press the Signed Decimal Display (F2) key, or press the ±DEC/R key on the 91-key keyboard. The number is now displayed in signed decimal form at:



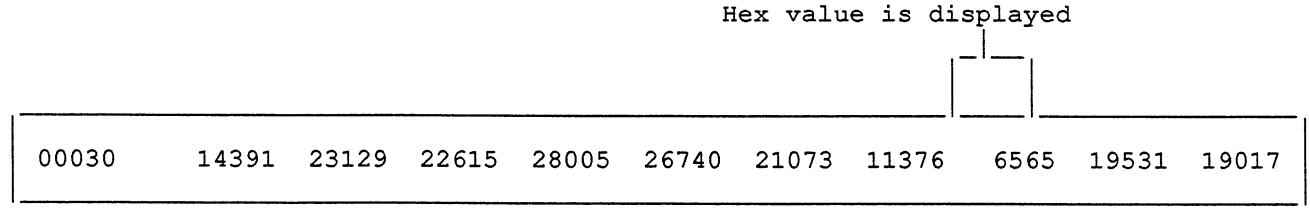
Displaying One Register in Hexadecimal Format

To display the contents of one register as hexadecimal values (OOOO to FFFF):

1. With the reference display on the screen, move the cursor to the register to be converted.



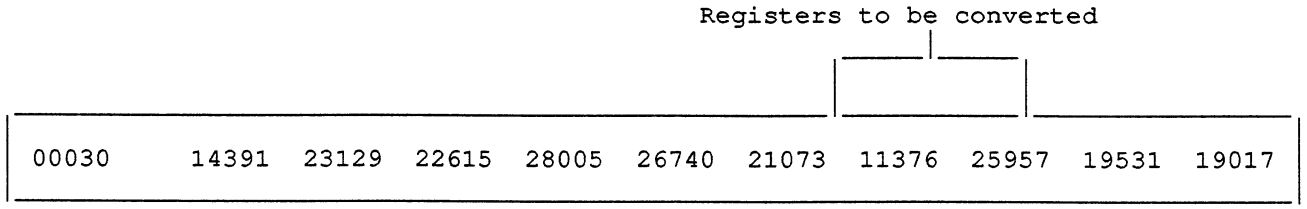
2. Press the Hexadecimal Display (F3) key, or press the HEX/I key on the 91-key keyboard. The number is converted to a hex value:



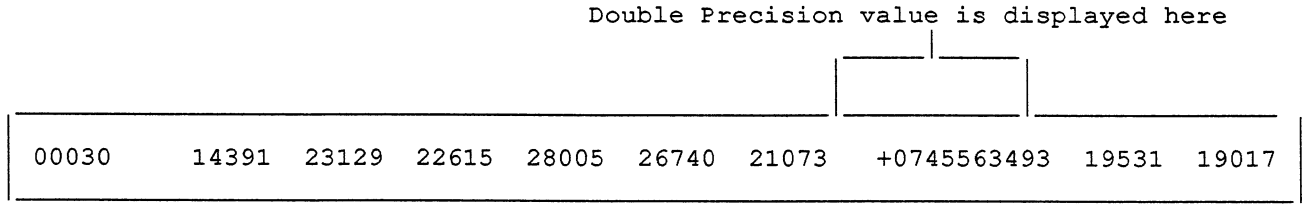
Displaying Two Adjacent Registers in Double Precision Format

To display the value in two adjacent registers as a signed double precision decimal number (-2,147,483,648 to +2,147,483,647):

1. With the reference display on the screen, move the cursor to the right register of the pair to be converted.



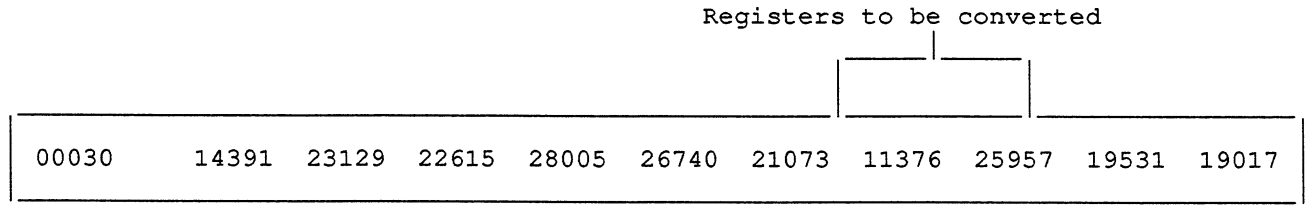
2. Press the Double Precision Display (F4) key. The value of the pair of registers appears in two's complement double precision format. If the sign bit is 0, the number is positive. If the sign bit is 1, the number is negative.



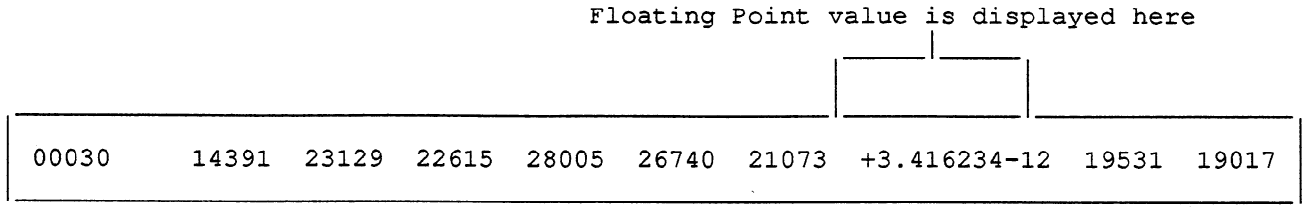
Displaying Two Adjacent Registers in Floating Point Format

To display the value in two adjacent registers as a floating point decimal number:

1. With the reference display on the screen, move the cursor to the right register of the pair to be converted.

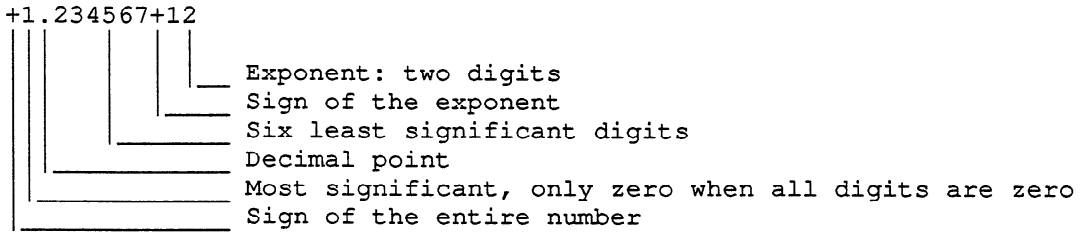


2. Press the Floating Point (F4) key. The value of the pair of registers appears in decimal scientific notation.



Format of a Floating Point Entry

The floating point value is displayed in the following format:



Displaying the Work Area in Floating Point Format

The Floating Point Display (F6) key changes only the format of the reference table display. It does not change the format of the work area.

To change the format of the work area and the element format at the cursor position to floating point, press the ALT and C/DP keys. To remove the floating point format, press the Clear key.

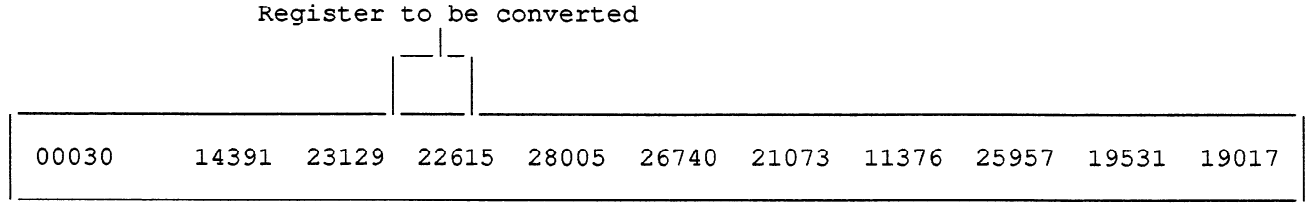
Press the following keys to change the base of the work area and the element at the cursor position:

Key	Description
Shift-Dec/0	Change to decimal format.
Shift-±Dec/R	Change to signed decimal format.
Shift-Hex/I	Change to hexadecimal format.
Shift-DP/C	Change to double-precision format.

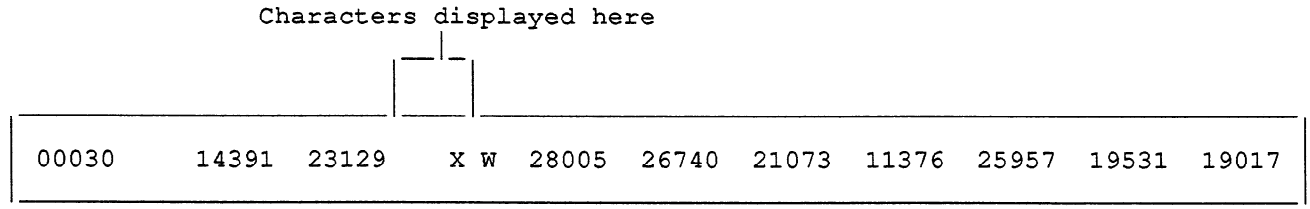
Displaying One Register in ASCII Format

To display the value of one register as two ASCII characters:

1. With the reference display on the screen, move the cursor to the register to be converted.



2. Press ALT-A. The 16 bits in the register at the cursor position (here, it is shown as in decimal) are divided into two sections, bits 16-9 and 8-1. Each section is represented as its ASCII character equivalent:



Bits 16 and 8 are parity bits and are ignored. The remaining 7 bits in each section are converted as shown in the table below. Command codes and non-displayable characters are appear on the screen as characters ^@.

ASCII Character Displays							
Bit Pattern	**Character**	**Bit Pattern**	**Character**	**Bit Pattern**	**Character**	**Bit Pattern**	**Character**
X0000000	^@	X0100000	(blank)	X1000000	@	X1100000	,
X0000001	^A	X0100001	!	X1000001	A	X1100001	a
X0000010	^B	X0100010	"	X1000010	B	X1100010	b
X0000011	^C	X0100011	#	X1000011	C	X1100011	c
X0000100	^D	X0100100	$	X1000100	D	X1100100	d
X0000101	^E	X0100101	%	X1000101	E	X1100101	e
X0000110	^F	X0100110	&	X1000110	F	X1100110	f
X0000111	^G	X0100111	'	X1000111	G	X1100111	g
X0001000	^H	X0101000	(	X1001000	H	X1101000	h
X0001001	^I	X0101001	)	X1001001	I	X1101001	i
X0001010	^J	X0101010	*	X1001010	J	X1101010	j
X0001011	^K	X0101011	+	X1001011	K	X1101011	k
X0001100	^L	X0101100	,	X1001100	L	X1101100	l
X0001101	^M	X0101101	-	X1001101	M	X1101101	m
X0001110	^N	X0101110	.	X1001110	N	X1101110	n
X0001111	^O	X0101111	/	X1001111	O	X1101111	o
X0010000	^P	X0110000	0	X1010000	P	X1110000	p
X0010001	^Q	X0110001	1	X1010001	Q	X1110001	q
X0010010	^R	X0110010	2	X1010010	R	X1110010	r
X0010011	^S	X0110011	3	X1010011	S	X1110011	s
X0010100	^T	X0110100	4	X1010100	T	X1110100	t
X0010101	^U	X0110101	5	X1010101	U	X1110101	u
X0010110	^V	X0110110	6	X1010110	V	X1110110	v
X0010111	^W	X0110111	7	X1010111	W	X1110111	w
X0011000	^X	X0111000	8	X1011000	X	X1111000	x
X0011001	^Y	X0111001	9	X1011001	Y	X1111001	y
X0011010	^Z	X0111010	:	X1011010	Z	X1111010	z
X0011011	^[	X0111011	;	X1011011	[	X1111011	{
X0011100	^	X0111100	<	X1011100	\	X1111100	|
X0011101	^]	X0111101	=	X1011101	]	X1111101	}
X0011110	^^	X0111110	>	X1011110	^	X1111110	.
X0011111	^_	X0111111	?	X1011111	_	X1111111	W

Displaying Register Contents as Text

The ASCII display shows the contents of registers as their ASCII character equivalents. If the registers from R00001 to R00013 contained the binary values (X)1100001 through (X)1111010, it would look like this on the ASCII display:

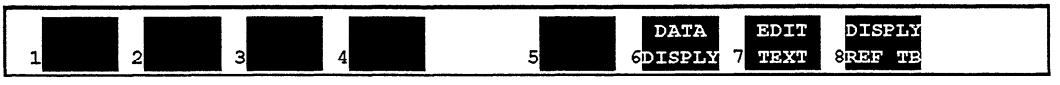
```
00010    t s    r q    p o    n m    l k    j i    h g    f e    d c    b a
00020  00000  00000  00000  00000  00000  00000  00000    z y    x w    v u
```

The lowest-numbered register appears at the right of the display, so the letters are not in normal sequence for reading.

The Text Display function reformats the entire screen. In Text Display format, the screen displays 16 registers on a line, beginning at the upper left. The registers above would look like this in Text Display mode:

```
00001    a b c d e f g h i j k l m n o p q r s t u v w x y z^@^@^@^@^@^@
00017   ^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
```

In Text Display mode, the bottom of the screen displays these function key assignments:



Typing in New Text Characters

New characters can be typed in from the keyboard while the screen is in Text Display mode.

To enter a single character, place the cursor at the character to be replaced, and type in the new character.

To enter a string of characters:

1. Place the cursor at the position where you want the first character to appear.
2. Press F7 (Edit Text).
3. Type in the new characters.
4. When the entry is complete, press the End Edit (F7) key.
5. Press F6 (Data Display) if you want to return to the register data display.

Converting All Registers to Another Format

To convert all the registers to the same format:

1. With the register reference display on the screen, press F7 (Change All).
2. Next, press the key(s) for the display type you want:

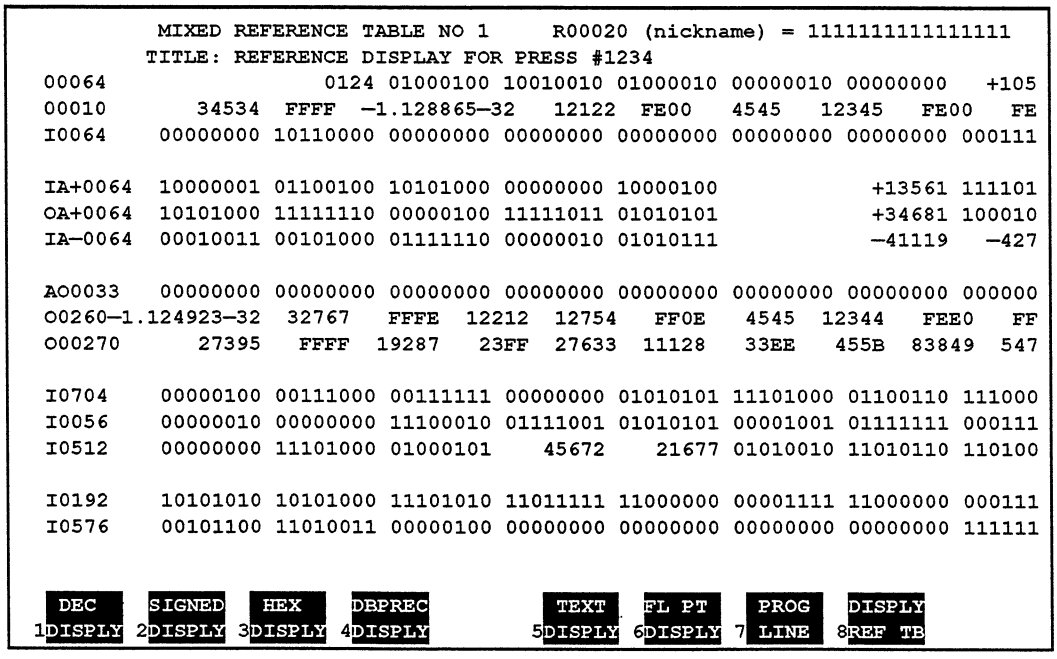
Function Key	Function	Description
F1	Decimal Display	Display values in decimal format.
F2	Signed Display	Display values in signed decimal format.
F3	Hexadecimal Display	Display values in hexadecimal format.
F4	Double Precision	Display values in double precision format.
F5	Text Display	Display values as ASCII characters, left to right.
ALT-A		Display values as ASCII characters, right to left.

Mixed Reference Tables

As many as eight screens combining discrete and register values can be created, each with a different format. These formats are used only for the mixed reference displays; they do not affect discrete or register table printouts.

Each Mixed Reference display has a number from 1 to 8. To access an existing or unused screen, type the screen number. Then, select Display Reference Tables (F3) from the Supervisor menu, or press F8 from a screen in the Display Reference Tables function.

Each line shows the values for one type of reference. It may be a line of ten register values, or a line of eight I/O values. The left column shows reference addresses.



The function key assignments on a Mixed Reference display will change depending on the location of the cursor. If the cursor is on a line of register values, the function key assignments shown in the sample screen above.

If the cursor is on a line of discrete values, the following function key assignments are available:

When the cursor is on a line of register values, the Floating Point Display (F6) key can be used to change the value at the cursor position. It does not change the format of the work area. To change the work area to floating point format for either register or discrete references, press the ALT and C/DP keys. To remove the floating point format, press the Clear key.

Creating a Title for a Mixed Reference Display

Each Mixed Reference display can have a unique title, consisting of up to 60 characters. To create a title, move the cursor to "TITLE:" on the display and type in the title for the display, using the ASCII keys. As you type, characters will appear from the right of the banner and move to the left. You can use space characters to position a short title where you want it to appear on the page.

Press CTRL-Z (or the Clear key) to delete all the characters you have entered. To delete just one character at a time, press CTRL-D (or the Delete key).

After entering the title, press the Down cursor key to move the cursor to the main area of the screen.

Changing or Adding a Line on a Mixed Reference Display

To change or add a line of discrete or register references, move the cursor to the line where you want the data to appear. It is not necessary to start at the top. Enter a reference from the new line in the work area, and press F7 (Program Line). A line of 64 discrete or 10 register values appears. The reference on the right side of a discrete line is divisible by 64 with a remainder of 1. The reference on the right side of a register line has a final digit of 0 on the left and 1 on the right (unless it is the last line of the register display). Continue to add or change lines as needed. Remember to move the cursor before making each new selection.

Register Displays on a Mixed Reference Display

In a Mixed Reference display, each line of register values begins with a reference address. The reference address is followed by ten values, which may be decimal, signed decimal, hexadecimal, double-precision, floating point, or ASCII. Two example registers are shown below.

```
00260-1.124923-32  32767   FFFE  12212  12742   FF0E   4545  12423   FEE0   FF
00270       27395   FFFF  19287  23355  12312  11224   33EE   455B  83849  527
```

Discrete Displays on a Mixed Reference Display

In a Mixed Reference display, each line of discrete values begins with letters that identify the I/O type, followed by the reference address. The reference address is followed by eight values, which may be decimal, signed decimal, hexadecimal, or binary. Two example registers are shown below.

```
00260-1.124923-32  32767   FFFE  12212  12742   FF0E   4545  12423   FEE0   FF
00270       27395   FFFF  19287  23355  12312  11224   33EE   455B  83849  527
```

Changing the Format of a Line in a Mixed Reference Display

When a line of discrete references first appears, it is in binary format. When a line of register values appears, it is in decimal format. You can change the format of a reference by following these steps:

1. Move the cursor to the item whose format you want to change. The function key assignments for discrete or register display appear at the bottom of the screen.
2. Press the appropriate key(s) to change the format. More information about these formats appears earlier in this chapter.

Discrete: Decimal (Decimal Display, F1)
Signed Decimal (Signed Display, F2)
Hexadecimal (Hexadecimal Display, F3)
Binary (Binary Display, F4)

Register: Decimal (Decimal Display, F1)
Signed Decimal (Signed Display, F2)
Hexadecimal (Hexadecimal Display, F3)
Double Precision (Double Precision Display, F4)
Text (Text Display, F5)
ASCII (ALT and A)
Floating Point (Floating Point Display, F6)

Making On-Line Changes

Single word changes can be made in Display Reference Tables mode. The effect of on-line changes depends upon the current mode (On-Line, Off-Line, or Monitor), and whether the program in the CPU is equal or not equal to the program in Logicmaster 6 system memory. For serial versions of the software, it also depends on whether on-line changes are enabled.

When the system is in On-Line or Monitor mode with an operating CPU connected, the information displayed in the reference tables is obtained from the CPU. This information includes I/O state and register content. Displays are updated as the CPU status changes.

When the system is in Off-Line mode, table values come from the program in Logicmaster 6 system memory. Power flow is shown with numerical values from internal memory.

WARNING

Improper use of on-line program changes can damage equipment or cause personal injury. Make on-line program changes with extreme care. On-line changes can have serious and unforeseen results on a control system if they are improperly used. These functions should not be used with people near the equipment. If possible, they should be done with direct visual control over the system. Proper external power disconnects should be made to prevent undesired equipment operation.

To make an on-line change, the status line at the top of the screen must show that the system is on-line to the CPU, and that the active program is exactly the same (''equal''). For example:

```
CPU: RUN/ENABLE    SWEEP:  7ms   L/M EQUAL CPU   L/M ONLINE  CURSOR: 000C
```

Making On-Line Changes, Serial Version

When making single-bit changes, the system reads the value of the bit on one sweep of the CPU, and writes back the command to make the on-line change on a later sweep. It is possible for the program or an external device to change the value of the bits of the byte or word before the Write command is received. If so, the value will then be incorrect.

Occasionally, with all types of on-line changes (bit, byte, or word) errors may occur because the system does not verify the change until a later sweep. Therefore, if an error has occurred, the system cannot try the change again.

On-line changes must be enabled in the Communications Set Up menu, as explained in a later section. If they are not enabled, the On-Line Change function key assignment will not be available.

NOTE

If the version of the CCM card firmware is earlier than version 5, program memory (not tables or registers) cannot be written to while the CPU is running.

If the version of the CCM card firmware is earlier than version 5, and the CPU was programmed using SCREQ commands, the system may encounter write-protected memory. If so, changes will not be permitted.

Changing Register Values

Register contents can be changed from the keyboard. If the system is on-line to the CPU, such changes are placed in CPU memory. The CPU Memory Protect switch can be in either the on or off position when changing register contents. If the system is off-line (not in Monitor mode), changes are loaded into the Logicmaster 6 system's internal memory.

A register can be loaded in decimal, signed decimal, hexadecimal, floating point, or double-precision format. To load a register, with the register table displayed on the screen, place the cursor on the register to be changed. The register will be updated by the value in the work area's numeric line.

Type in the value for the register using the numeric keypad. The entry appears in the work area. When the entry is correct, press CTRL-E (or the Enter key) to load the value into the register.

Using Overrides in I/O Tables

Discrete I/O and Auxiliary I/O references can be overridden from the Display Reference function using the Override Reference (F5) or Toggle (F6) function key. In a system that uses Expanded addressing, I/O assigned to channels 1-7 and 9-F cannot be overridden. Therefore, inputs and outputs for which overrides will be used must be assigned to the main and auxiliary I/O tables.

An override removes control of the reference from its normal source. Overridden inputs ignore information from the devices wired to the I/O structure, such as limit switches or pushbuttons. Similarly, overridden outputs ignore programmed logic and internal power flow. Non-relay functions such as timers, counters, arithmetic functions, and moves, still work when a coil is overridden.

Overrides are retained even when power is removed from the system. The ladder logic cannot change overrides; however, non-relay functions such as timers, counters, arithmetic functions, and moves can change the state of an overridden reference.

The override is a very powerful tool for program checkout and maintenance. You can test a program in a PLC that is not connected to an I/O structure by using overrides to simulate inputs. You can also check out a program when I/O is connected, by using overrides to prevent coil operation.

After the I/O is wired up, it can be tested by activating each coil with an override to verify I/O communications, module operation, power to a device, wiring to a device, indicator lights, fuses, and other hardware.

After the control system is thoroughly checked out and placed in operation, the override is very useful in a monitored system. If a sensor or input module should fail while the process is in operation, that input can be overridden. Thus, the process can be continued until it can be shut down safely.

Using the Display Reference Function to Identify Overrides

References should not be overridden when the Logicmaster 6 system is removed from the process, or when making copies of a program. Use the Display Reference Tables function to verify all inputs and coils before removing them from the programmer, or copying the program.

Using Overrides

Overrides should be used on an operating system only with extreme care.

WARNING

Improper use of the override can damage equipment or cause personal injury.

To use an override, the CPU Memory Protect switch must be in the OFF position.

1. Display the area of the program you want to change and press the On-Line Change (F4) key. The screen shows the new function key assignments.
2. Place the cursor on the reference to be overridden.

CAUTION

The reference will be overridden throughout the program, not just at the cursor location.

3. Press F4 (Override) to toggle the state of the reference between overridden and not overridden.

Forcing the Status of a Reference

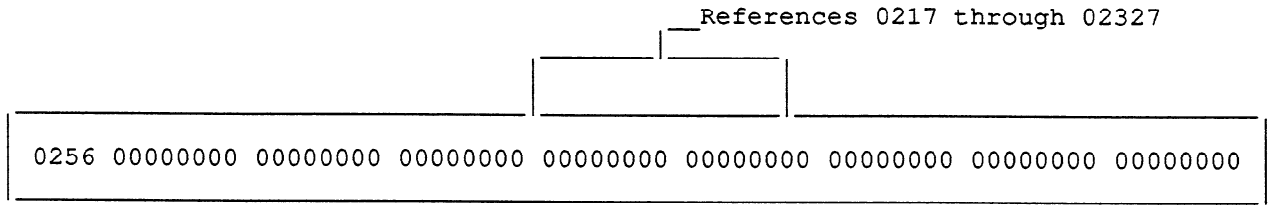
In the Display Discrete References function, the Toggle Reference (F6) function key is used to toggle the status of a reference between on and off. If the reference being forced is currently overridden, it retains its new status until the Toggle Reference key is pressed again. If the reference is not overridden, it retains its new status until changed by some other function, such as rung solution or I/O servicing. This usually occurs within one scan.

To toggle a reference:

1. With the table displayed on the screen, place the cursor on the reference to be forced.
2. Press F6 (Toggle Reference) to change that reference to its opposite state. All logic elements in the program that use the reference will reflect the new status.

Changing the Value of a Register of Discrete References

The Toggle Reference key can only be used to change the status of one discrete reference at a time. You can also change the value of the entire register where the cursor is located. To do that, first you must change the number base of the register. For example, suppose you wanted to convert discrete references (see below) 0217 through 0232 to all ones.



Follow the steps below to change a register of discrete references.

1. Be sure the cursor is in the rightmost (lower numbered) byte of the register to be converted. Use the function keys to change the number base to one of the following: decimal (by pressing F1), signed decimal (F2), or hex (F3).
2. If the work area cursor is not on the bottom line, press CTRL-S (or the Select key) to move it there.
3. In the number base you chose above, type in the equivalent of the binary value you want to enter.
4. Press CTRL-E (or the Enter key) to change the reference table value at the cursor position to the new value.
5. Press F4 (Binary Display) to convert the base back to binary values.

Chapter 8
Print Functions

The Print functions can be used to print copies of ladder logic and annotation. You can print all of a program, or just a part of it. The program currently in RAM or the content of the screen can be printed in foreground mode. During a foreground mode print, the system is dedicated to printing, and cannot be used for anything else. A program stored as a print file can be printed in background mode, allowing the system to be used for other functions at the same time.

Printing Programs Created Prior to Release 3

The first time you use Release 3.01 or later software to print an older program, Logicmaster 6 converts the structure of the .NAM, .EXP, and .RDF files, making them incompatible with earlier versions of the software.

You should first copy these files onto a newly-formatted program diskette, and use these copies. If a disk error should occur while the files are being converted for printing, copy only the .NAM, .EXP, and .RDF files from the diskette and try again.

Printing a Copy of the Screen

For quick reference, the Print Screen function can be used to print a copy of the current screen display. Print Screen will print 25 lines of 80 characters.

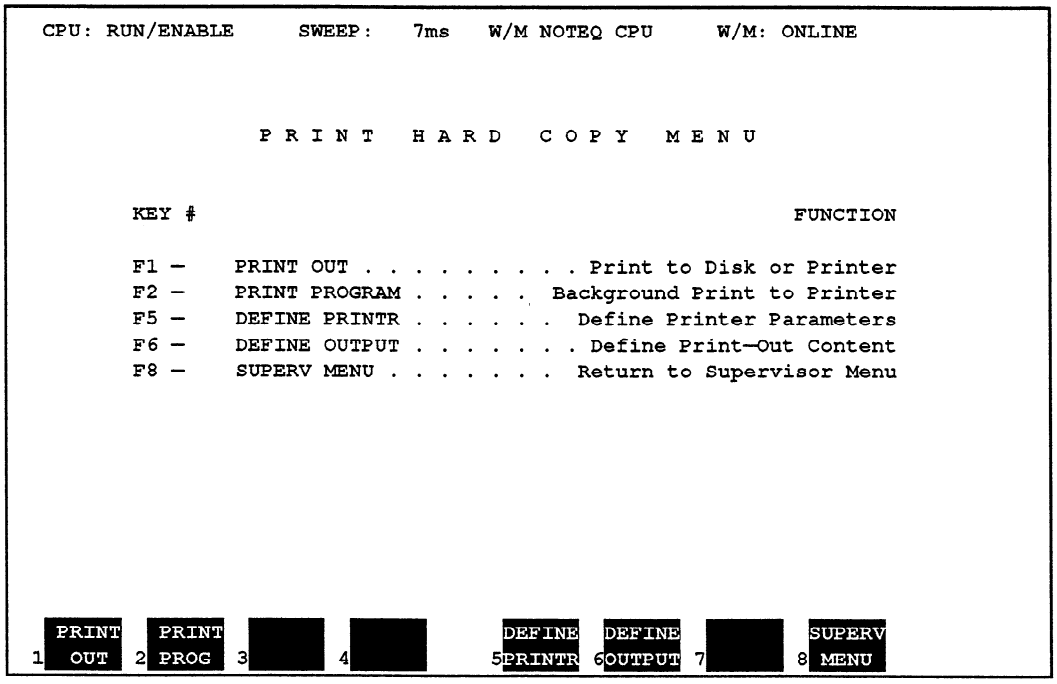
Before this function can be used, the printer must be set up for printing in background mode. After this is done, the contents of any screen can be printed out. To execute a screen print, with the screen displayed, press ALT-P. The printout stops when the screen has been printed. To stop the printout sooner, press any other key.

Displaying the Print Menu

When Print function (F5) is selected from the Supervisor menu, the Print menu is displayed.

NOTE

For Logicmaster 6 software on 5 diskettes, the Print functions are on diskette 3. If this diskette is required, the system will prompt you to insert the Overlay disk.



Function Key	Function	Description
F1	Print Out	Send the program to either a printer or a diskette. The Print Out function is done in foreground mode. Use Print Out to store a program on a diskette for printing with the Print Background function.
F2	Print Program	Print a program from a print file on a diskette. The printing is done in background mode, so the system can be used for other activities while the program is printing out.
F5	Define Printer	Specify the print format to be used with the printer.
F6	Define Output	Specify the content of printouts (for example, with annotation or cross-references).

Attaching a Parallel Printer

1. Attach the printer to a parallel printer port. With a Workmaster computer, use the lower port on the Combination Adapter card.
2. When you are ready to use the printer, turn it on and place it in On-Line mode.
3. To use the Print functions, you must define the printer parameters, as explained on the following pages.
4. If you expect to use the Print Screen command, set up the printer for printing in background mode by pressing F5 (Print Functions) from the Supervisor menu and then pressing F2 (Print Program) from the Print menu. Enter the number of the destination port; for the Workmaster computer, this is port 3.

The printer may now be used with the Print Screen command.

Attaching a Serial Printer

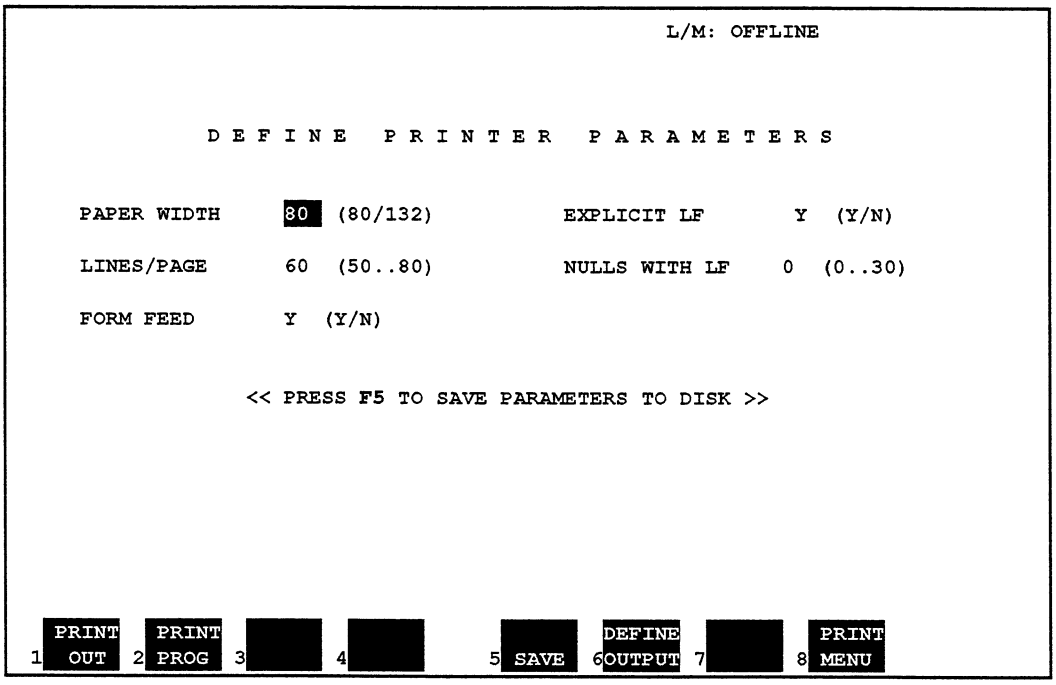
1. Set the DIP switches and jumpers on the printer, as instructed in the manual for the printer.
2. Attach the printer to serial port 1 or 2. With a Workmaster computer, the upper port on the Combination Adapter card is port 1.
3. In the Supervisor menu, select Utility Functions (F8). Then, press F6. Enter the parameters for the port (1 or 2) which correspond to those set up for the printer. Press F1 to store the printer parameters.
4. When you are ready to use the printer, turn it on and place it in On-Line mode.
5. To use the Print functions, you must define the printer parameters, as explained on the following pages.
6. If you expect to use the Print Screen command, set up the printer for printing in background mode by pressing F5 (Print Functions) from the Supervisor menu and then pressing F2 (Print Program) from the Print menu. Enter the number of the destination port; for the Workmaster computer, this is port 1 or 2.

NOTE

Do not disconnect the printer cable while printing. When the cable is reconnected, the printout will be incorrect.

Defining Printer Parameters

The first time a printer is used with the system, its print output format must be defined so the system can format the data properly. To define the printer parameters, press F5 (Define Printer) from the Print menu.



To change entries on this screen:

1. Move the cursor to the item you want to change.
2. Enter new values as appropriate. If you need more information, refer to the definitions on the following pages.
3. When you are finished, decide whether your entries or the default values shown above should be used when the system is started up.

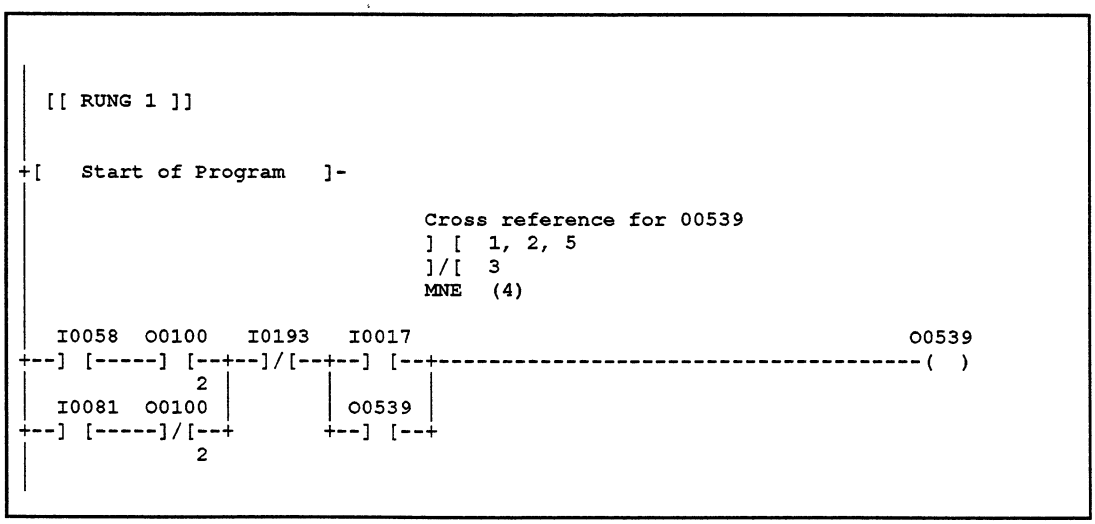
   A. To save the printer parameters only until the next power-up and then return the default parameters, press F8 (Print Menu) to exit.

   B. To save the printer parameters until this screen is changed again, press F5 (Save) to exit. This causes the printer parameters to be written to the PRINTER.SET file. For a floppy diskette system, this file will be written to drive A. A write-enabled master or copy diskette must be present in that drive. When the Logicmaster 6 system is powered up, these parameters replace the standard default values.

Paper Width: (80/132)

The entry for paper width determines the number of characters that will be printed on a line. If the printer is set up for standard 8-1/2 inch wide paper, select 80 characters for the paper width. If the printer uses 11-inch (or wider) paper, you may select 132 characters.

In 80-character mode, the cross-references for a rung are printed above that rung.



In both 80-character and 132-character mode, the ladder diagram itself is printed in the same way. However, in 132-character mode, coil labels are printed at the right side of the rungs, and cross-reference information in the tables is spread over 132 characters instead of 80.

Number of Lines per Page: (50..80)

The default value for the number of lines to be printed on a page is 60. This can be changed to from 50 to 80 lines per page.

If an associated group of lines will not fit on one page, the system will command the printer to advance to the next page after the number of lines specified by this entry. Note that some printers automatically insert a form feed after printing a certain number of lines (typically, 66). If the printer has this feature, specify a shorter page length to prevent an automatic page eject.

Form Feed: (Y/N)

This item determines whether the system automatically inserts an ASCII form feed character at the end of a page. This item should be set to **Y** if the printer recognizes a form feed. If the printer does not recognize the form feed character, set this item to **N**. That will cause the system to insert a sequence of carriage returns to advance to the top of the next page.

Line Feed: (Y/N)

The line feed character advances the paper to the next line for printing. This item determines whether the system automatically inserts a line feed character each time the printer head returns to the left page margin. This item should be set to **Y** to have the system insert a line feed character after each carriage return character. Set this item to **N** if the printer automatically advances the paper each time it encounters a carriage return character.

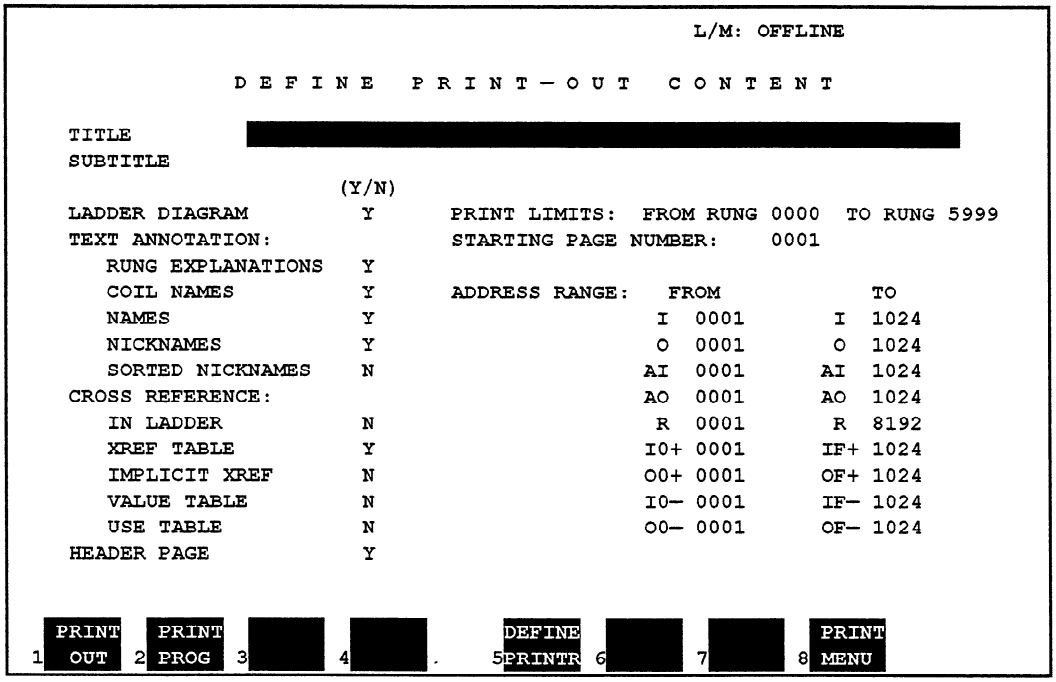
Nulls with Line Feed: (0..30)

This entry is usually set to 0. If a value is entered here, the system will insert that number of null characters after each carriage return character. Some printers require these null characters to give the print head time to return to the left margin after the carriage return.

Displaying the Printout Content Screen

The first step in defining the content of the printout is to display the Printout Content screen. If you want to print annotation, enter the program name from the Supervisor menu before displaying the Print menu.

In the Print menu, press F6 (Define Output). If a program name was not entered, the annotation choices default to **N** and you cannot change this to **Y**.



To change the entries on this screen:

1. Move the cursor to the items you want to change.
2. Type in new entries. If you need more information, refer to the explanations on the following pages.

NOTE

To reduce the time needed for a cross-reference printout, select only the address range actually needed. Setting the system mode to Off-Line (using the Mode Select keyswitch, or on the Scratch Pad screen), should also reduce cross-reference printout time.

3. When the entries are complete, press F8 (Print Menu) to return to the Print menu. If the printer parameters have been established, the content can be printed using the Print Out (F1) function.

Title

To print a main title at the beginning of each page, enter the title here. This title can be changed with the rung explanations during the printout, as explained in chapter 6, *Annotation*. To enter a title:

1. If the reverse-video banner does not appear at the top of your screen, press the Up cursor key as needed.
2. Type in a title of up to 60 characters. To make changes, press CTRL-D (or the Delete key) to delete the last character, or press CTRL-Z (or the Clear key) to delete all the characters at once.

Subtitle

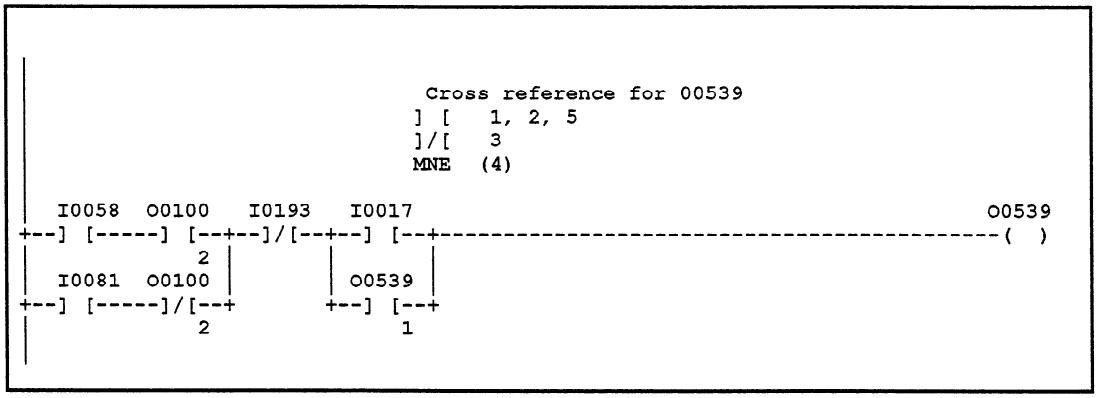
To print a subtitle below the title on each page, enter it here. The subtitle is entered like a title, as explained above.

Ladder Diagram: (Y/N)

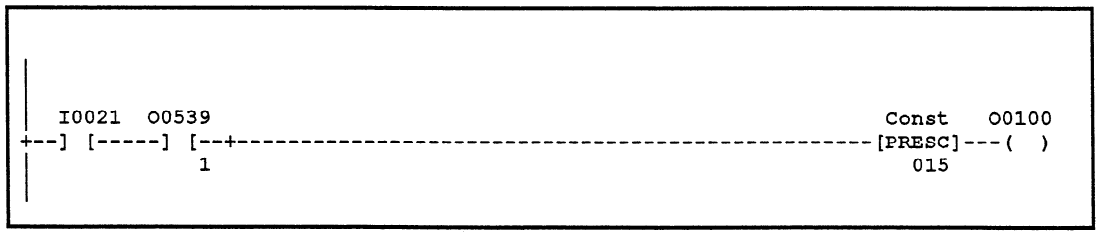
Select whether or not the printout will include any of the ladder logic.

Ladders are printed by rung numbers. Page breaks occur between rungs, or between the cross-reference list and the rung in 80-character mode. Ladder logic is printed as up to nine horizontal elements, up to eight parallel lines of logic. The coil is printed to the right, in the tenth column.

If In-Ladder Cross-Reference is selected, under each relay contact that is referenced to a coil, the printout shows the number of the last rung where the scan encountered that reference as a coil.



In the ladder, constants are printed with their decimal values.



After the ladder diagram is printed, if explanations or labels have been requested, additional explanations and coil labels for which there are no program rungs will be printed. This will show whether the program has additional unnecessary annotation.

Print Limits: (From/To Rung)

Enter the rung in the program where you want the ladder diagram printout to begin, and to end. Include any leading zeros.

If the program contains a Configuration function, it will be printed out as rung 1. Above the rung, the printout includes a copy of the CPU Configuration Setup menu and the Genius Bus Controller Locations menu. These menus show the current setup of the CPU and the Genius I/O.

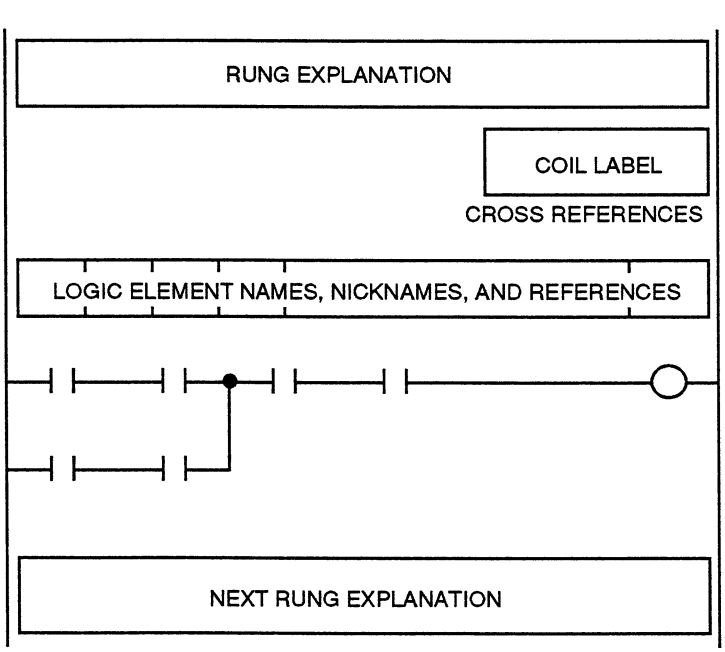
Starting Page Number

To begin numbering the printout with a page number other than 1, enter the number here.

Text Annotation: (Y/N)

Select whether to print any annotation. To change the default selections, enter **Y** or **N**.

The illustration below shows the basic format for an 80-column printout with annotation. A 132-column printout is the same, except that the coil label and in-ladder cross-references, if printed, appear to the right of the ladder diagram.



If either rung explanations or coil labels are requested without the ladder diagram, then just the annotation is printed. In that case, any control characters for titles, subtitles, borders, or side files are printed out (instead of executed as they would be in a ladder diagram printout).

Sorted Nicknames: (Y/N)

Select whether to print a list of all the nicknames used in the program, sorted alphabetically. If the selection for names is **Y**, then the names will also be printed.

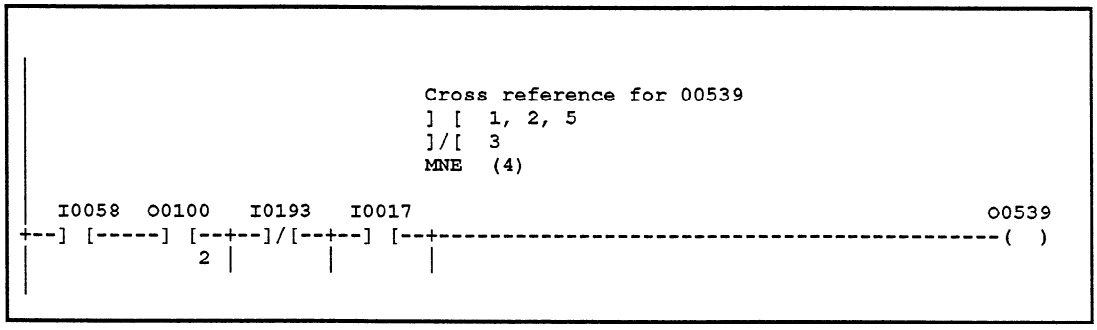
Cross References: (Y/N)

Print cross-references with or without printing the ladder diagram. To change the default selections, enter **Y** or **N**.

Before the cross-references can be printed out, the system must first create temporary files containing the selected cross-references. These files are placed on the diskette containing the active file. If no active file name has been specified at the Supervisor menu, the files will be placed on the disk in the default drive.

In Ladder: (Y/N)

Select whether or not the ladder diagram printout will include cross-references for coils. These are placed above the rung in 80-character mode, and beside the coil in 132-character mode. For example, the 80-character mode has this format:



Xref Table: (Y/N)

Select whether to print at the end of the ladder diagram a listing of all the references that appeared in the printout. Note that printing cross-reference tables takes a considerable amount of time.

Discrete References Cross-Reference Printouts: This listing groups discrete references into normally-open contacts, normally-closed contacts, the various types of coils, and non-relay (mnemonic) functions. The left column of the discrete references printout lists the references in ascending numerical order. Under the reference, the override status of either an I/O or Auxiliary I/O reference may be displayed. To the right, all occurrences of that reference in the program are listed by rung number. For example:

```
          *****  I N P U T  S T A T U S  T A B L E  *****

INPUT/              CROSS REFERENCES
(OVR)

I1004         :     ] [ 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27
                     28, 29, 30, 31, 32, 33, 34, 35, 36
                    MNE 88, 92, (199), 204, (286)

I0013         :     ]/[ 13

I0100         :     ]/[ 14
[Ovr On]

I0101         :      ] [ 108

I0605         :      ] [ 114
[Ovr Off]           MNE 100, 101, 101, (102), 105, (150), 200

       ( ) = implicit references
      MNE = mnemonic references
```

Register Cross-Reference Printouts: Register cross-reference printouts list the references on the left side. Under each reference, its value appears in the format that was selected with the Display Reference Tables function. To the right, all occurrences of that reference in the program are listed by rung number. For example:

```
      *****  R E G I S T E R  S T A T U S  T A B L E  *****

REGISTER/           CROSS REFERENCES
VALUE

R00004        :       11, 112, (117), 143, 186, 192, 193, 195, 199, 200
[00886]               229, 229, 229, 230,       32, (337), 433, 535, 647

R00013        :       13
[+01023]

R00100        :       147
[-00008]

R00102        :       140
[D.P.]

       ( ) = implicit references
```

Implicit Xref: (Y/N)

Select whether the printout will also include implicit references (for example, references included in tables, matrices, convert functions, and similar multiple reference functions). Rungs that include only implicit references are enclosed in parentheses, as shown in the example printout above.

The maximum number of implicit references that will be printed out for one reference is 1000.

Value Table: (Y/N)

Select whether to print the values of all references at the end of the ladder diagram. Reference printouts have the format set up with the Display Reference Tables function.

Discrete references are printed as individual pages of each type of reference (I/O, or Auxiliary or Expanded I/O). The override status of I/O and Auxiliary I/O references is also printed as individual pages. The partial example below shows discrete references printed out in binary format.

```
        *****  A U X  I N P U T  V A L U E  T A B L E  *****

AUX IN  |64       |48       |40       |32       |24       |16       |8       |1
ADDRESS +-------- +-------- +-------- +-------- +-------- +-------- +-------+

  0064  01101011 11001011 10010100 11011010 11110000 00001000 00110100
  0128  01010101 00100100 11101011 01010101 10100101 10101001 10101100
  0192  11110100 00100100 00000000 01001000 11111010 11010001 10100010
  0256  10100110 10100101 10100100 10101000 10101011 11001011 10101000

  0320  10101010 10100110 10100101 10101001 11100111 10100101 10100110
  0384  10101100 10010101 10010101 10010101 10010101 10010000 00100100
```

The next partial printout includes registers that have been reformatted using the Display Reference Tables function.

```
          *****  R E G I S T E R  V A L U E  T A B L E  *****

REGISTER |10     |9    |8    |7     |6     |5     |4     |3     |2     |1    |
ADDRESS  +------ +---- +---- +------ +------ +------ +------ +------ +------ +----+

00010 +00003 -00345 +00000 +23233 +00000 +00000   11D6 +00000 +00000 +01233
00020   +0023566777 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +01001
00030 +00000 +00000   01C8 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00040 +00000  23446 +00000  +3.416234-12 +00000 +00000 +00000 +00000 +00000
```

Used Tables: (Y/N)

The References Used tables show how references are used in a program. This is a partial example:

```
          *****  A U X  O U T P U T  U S E D  T A B L E  *****

AUX OUT  |64      |56      |48      |40      |32      |24      |16      |8
ADDRESS  +------- +------- +------- +------- +------- +------- +------- +-----

AO0064   **---*** **--*--- RRRRRRRR RRRRRRRR -----*** ***-**-- RRRRRRRR RRRRRR
AO0128   -------- -------- -------* **-*---- RRCRRRCR RRCRRRRC -------- --****
AO0192   -------- -------- -------- -------- -------- -------- -------- ------
AO0256   **--*--- ----**** -------- ---***** *******- -------* ****-*-- ------
```

The following symbols may appear in the Reference Used tables:

Symbol	Description
C	There are many ways to reference the same physical point. For example, AO0017, a program function using AO0001 with a length of 32 points, and a program function using R0001 with a length of 2 registers all reference the same point. It is important not to use the same point for multiple references. The letter C in the Use tables means that the reference is also used by another reference in the program.
E	A register or expanded reference's storage is used by the Expanded I/O in at least one of its 16 bits.
A	A register or expanded reference's storage is used as an auxiliary reference in at least one of its 16 bits.
R	An auxiliary reference's register storage is used as a register reference.
+	The reference is used as an implicit reference.
*	The reference is used as an explicit reference.
-	Not used; the reference is available for future use.

The symbols in this table are listed in order of priority. The highest priority symbol is printed. Appendix D, *Reference Used Tables*, shows the possible types of physical point usage, and the character for each.

Address Range: (From/To)

Printouts of discrete and register references default to the entire table of values. To print a smaller range of values, move the cursor to the type of reference and enter the address range. The value entered for "From" must be lower than the value entered for "To."

Header Page: (Y/N)

Select whether to include a header page at the beginning of the printout. The header page has the title and subtitle entered on the Define Print-Out Contents screen. It displays information about the program.

```
**************************************************************************
*                                                                        *
*                                    LAST REFERENCE USED:                *
*                                    ------------------                  *
*        INSTRUCTION SET: ADVANCED            REGISTER:    R16384       *
*                CPU ID:        1               INPUT:     I0001       *
*        CPU MEMORY SIZE:    32768              OUTPUT:     O0333       *
*    PROGRAM MEMORY SIZE:    32760           AUX INPUT:    AI1024       *
*   REGISTER MEMORY SIZE:    16384          AUX OUTPUT:    AO1024       *
*                                      EXP STATE  INPUT:   I9+0032       *
*                                                OUTPUT:   O4+1020       *
*                                      EXP STATUS INPUT:   I2-0654       *
*                                                OUTPUT:   OF-1001       *
*                                                                        *
**************************************************************************
```

If any type of cross-reference is selected, In-Ladder, Xref table, or Use table, then at the right side of the box a list shows the highest reference used in each type.

Each subsequent page starts with a line showing the current date and time. Under that, the title and subtitle appear.

Printing in Foreground Mode

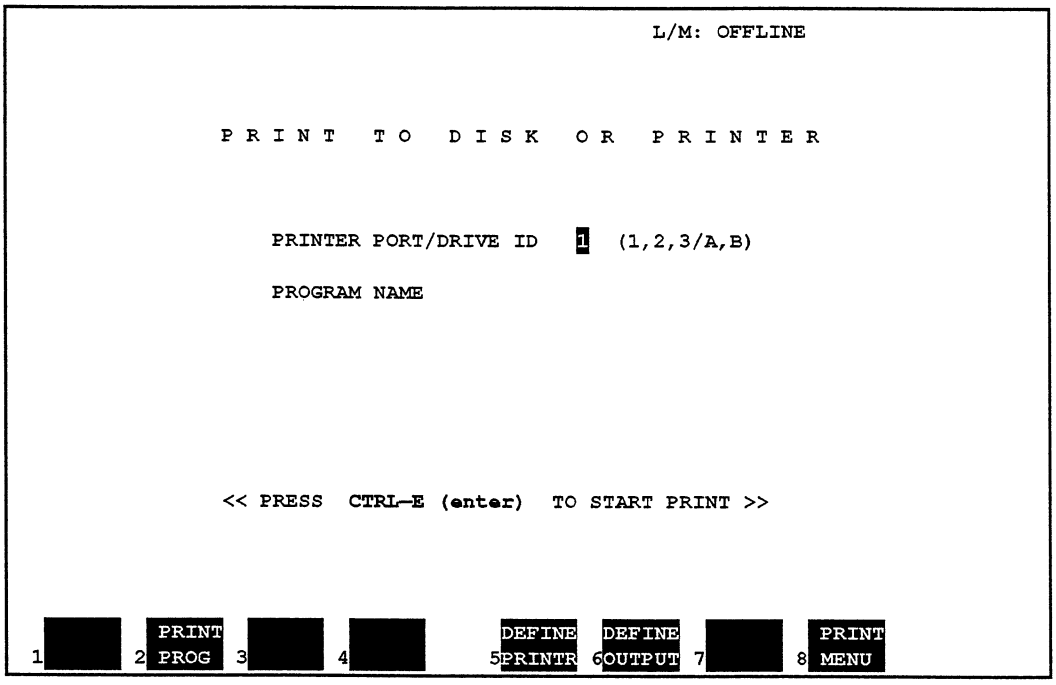
The Print Out function executes in foreground mode to print the program currently in Logicmaster 6 system memory. While the program is printing, however, the system cannot be used for anything else.

NOTE

For shortest printing time when printing in foreground mode, the Logicmaster 6 system should be placed off-line.

Printing out a complete program can take a considerable amount of time. It is often preferable to first store the information in the form of a print file. The file can then be printed out later in the background mode. The Print Out function can be used to store the program as a print file.

To print or create a print file in foreground mode, press F1 (Print Out) from the Print menu.



To send the print output directly to a printer for immediate printing in foreground mode:

1. If you are sending the output to a printer, be sure the printer characteristics have already been defined.
2. With the cursor at "Printer Port/Drive ID," enter the destination for the printout:
   A. Enter number **3** if the printer is connected to the parallel port.
   B. Enter number **1** or **2** if the printer is connected to a serial port, depending on the number of the serial port.
3. Press CTRL-E (or the Enter key) to start the printout.

Creating a Print File

To send the program to a disk drive for later printing in background mode:

1. For a floppy-diskette system, be sure a write-enabled diskette is in the drive.
2. With the cursor at "Printer Port/Drive ID," enter the letter destination of the drive to receive the file.
3. Move the cursor to "File Name" and enter the name for the printout.

   The first text file created has the file name extension .TXT. If the diskette runs out of space while the file is being stored, the screen prompts for another diskette to be inserted. This second part of the print file has the extension .TX1. For example, PROGRAM1.TX1. Subsequent portions of the same file will have the file name extension .TX2, and so on.

   NOTE

   If the text file is being created on the same disk containing the annotation or temporary cross-reference files, the text file may not be continued beyond one disk. Therefore, if you have a dual-drive system it is recommended that you always print to a drive that does not contain the annotation files or temporary cross-reference files.

4. Press CTRL-E (or the Enter key) to begin creating the file.

Stopping or Canceling a Printout

To temporarily stop printing, press the Pause Print (F4) key. To resume printing at the place it was stopped, press F4 again.

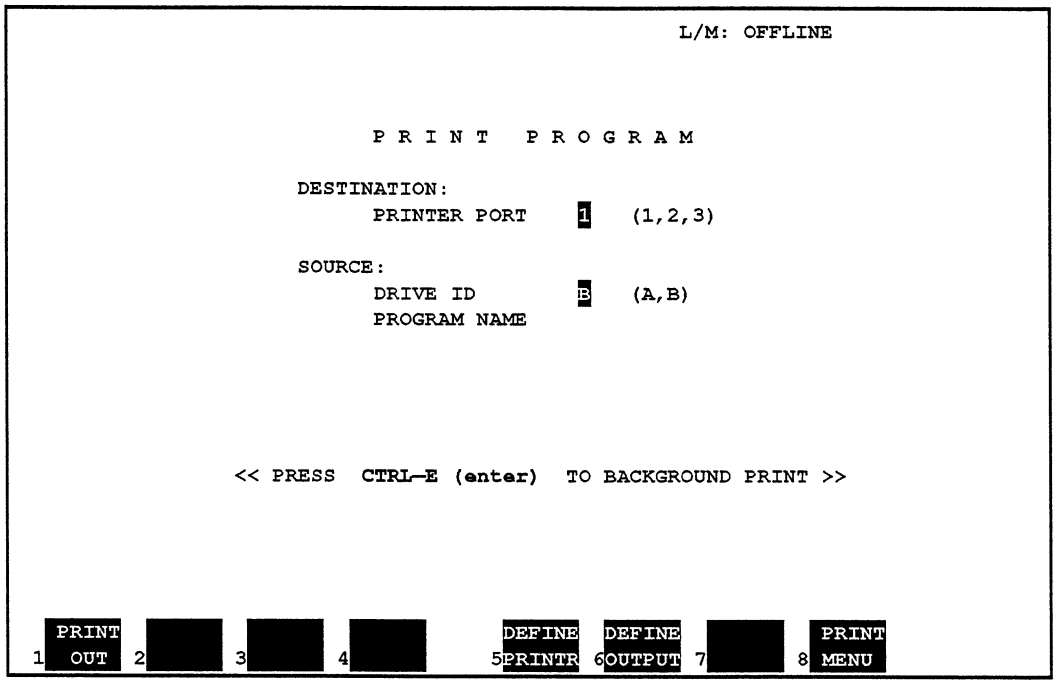
To end a printout in progress, press the Abort (F3) key and respond to the confirmation prompt. Aborting the printing stops only the printing; it does not delete the print files that were used for printing. These files must be deleted using the Utility functions.

Printing in Background Mode

The Print Program function prints a file from disk. Any file may be printed, not just the .TXT files that were created with foreground print.

Print Program is a background function; the system can be used for other activities while the printout is in progress. With the serial version of Logicmaster 6 software, if a screen is being printed using the Print Screen command ALT-P, printing terminates if the operating mode is changed from Off-Line mode to another mode.

To print out a print file, select Print Program (F2) from the Print Hard Copy menu.



To send the print file to a printer port for printing in background mode:

1. Be sure that the printer is properly connected, and that it is turned on.
2. With the cursor at "Destination: Printer Port," enter the destination for the printout:
   A. Enter number **3** if the printer is connected to the parallel port.
   B. Enter number **1** or **2** if the printer is connected to a serial port, depending on the serial port.
3. Move the cursor to "Drive ID" and enter the letter designation of the drive where the print file is stored.
4. Move the cursor to "Program Name" and enter the file name and extension. This is usually a .TXT file created with foreground print. However, any text file may be printed.
5. Press CTRL-E (or the Enter key) to start the printout. The screen displays the message "BACKGROUND PRINT IN PROGRESS."
6. Press F8 to return to the Print menu.
7. To end a printout in progress, press F3 (Abort) and respond to the confirmation prompt.

Chapter 9
Load/Store/Verify

The Load/Store/Verify functions are used to transfer programs and tables to and from the system's RAM memory. They are also used to compare the data in RAM memory with the stored data, and to clear RAM memory.

NOTE

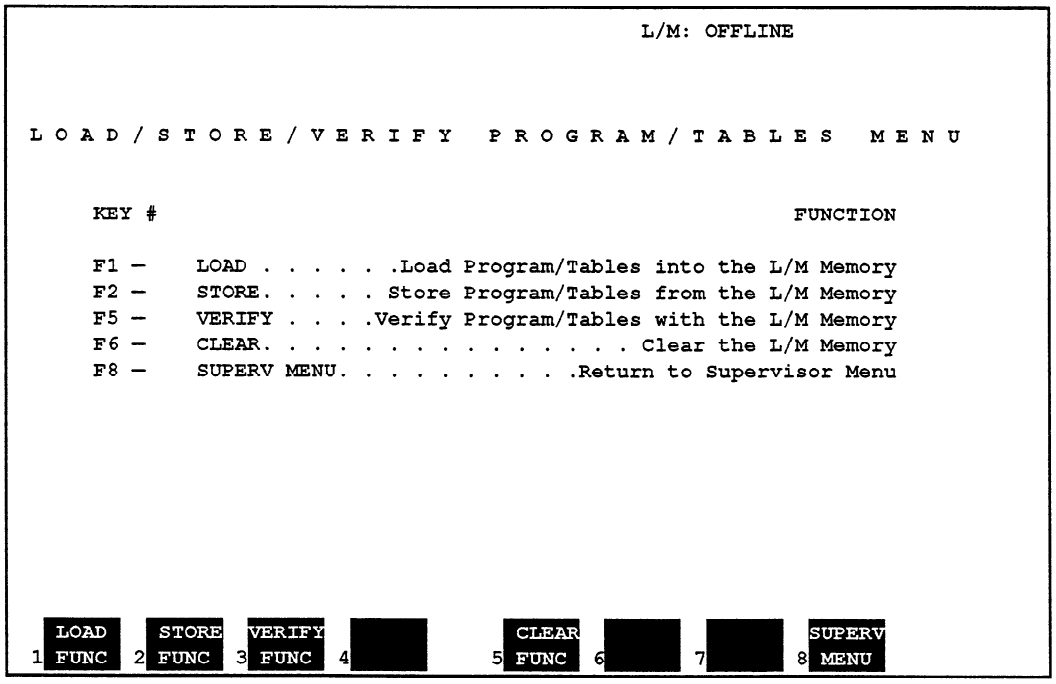
For serial versions of the Logicmaster 6 software, CCM2 revision E firmware or CCM3 revision C firmware is required to communicate with the CPU.

Logicmaster 6 software uses the operating mode (keyswitch setting) to determine what information will be transferred to the CPU, to disk, or to Logicmaster 6 memory.

To/From	Load/Store	Operating Mode		
		Off-Line	Monitor	On-Line
CPU	load	ladder and ref tables	ladder only	ladder only
	store	ladder and ref tables	not available	ladder only
disk	load/store	ladder and ref tables	ladder and ref tables	ladder and ref tables

Displaying the Load/Store/Verify Menu

When Load/Store/Verify (F6) is selected from the Supervisor menu, the Load/Store/Verify menu is displayed:



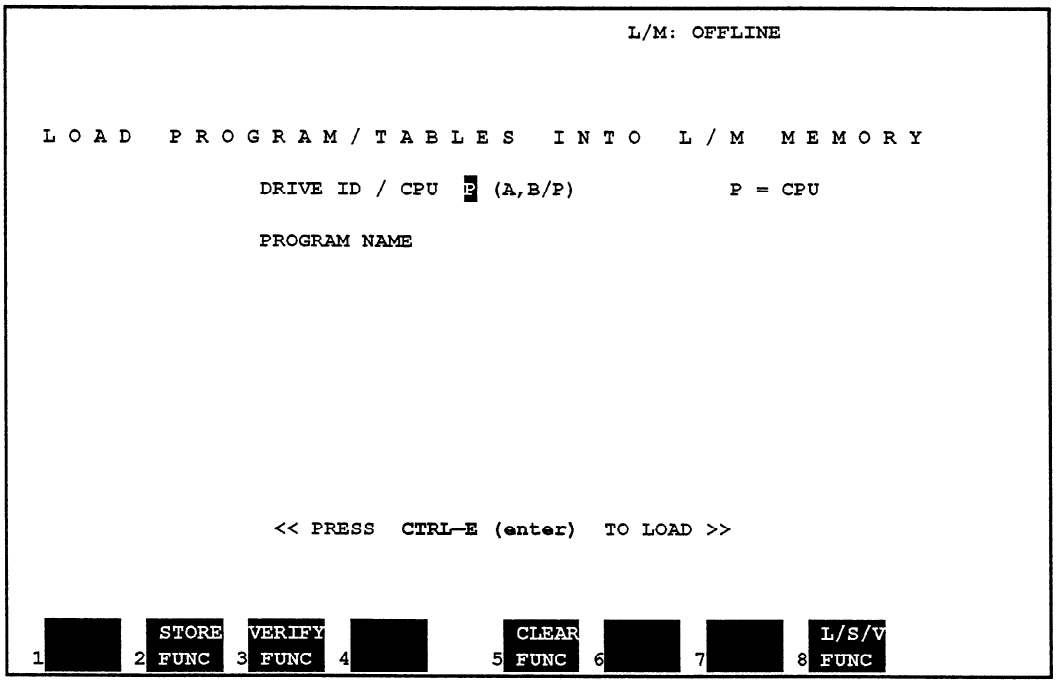
Function Key	Function	Description
F1	Load Function	Copy the program, and the register and I/O tables into RAM memory. This writes over any program or data already in RAM memory.
F2	Store Function	Copy program and table data from RAM memory to a drive, or to the CPU.
F3	Verify Function	Compare data in RAM memory with data in the CPU, or on a disk drive. A listing of differences can be printed out.
F5	Clear Function	Remove the program, and register and I/O tables from RAM memory.

Loading a Program into System RAM Memory

The Load function transfers a program and its associated tables into RAM memory from external storage, or from the CPU. The Load function copies the program, which remains unaltered in its original location.

If the Logicmaster 6 system is in Off-Line mode, the entire program including the ladder logic and tables, is copied into RAM memory. In On-Line or Monitor mode, only the ladder logic is copied.

To use this function, select Load Function (F1) from the Load/Store/Verify menu.



CAUTION

Copying a program into RAM memory will destroy a program already stored there. If there is program data in RAM memory, it can be stored to another device with the Store function before loading in the new data.

To copy the program into RAM memory from a storage device, or from a CPU:

1. With the cursor at "Drive ID/CPU," enter the letter that represents the disk drive where the program is stored, or the letter **P** for the CPU.

CAUTION

Do not change the CPU operating mode from RUN to STOP while loading a program from the CPU. This causes a loss of communications.

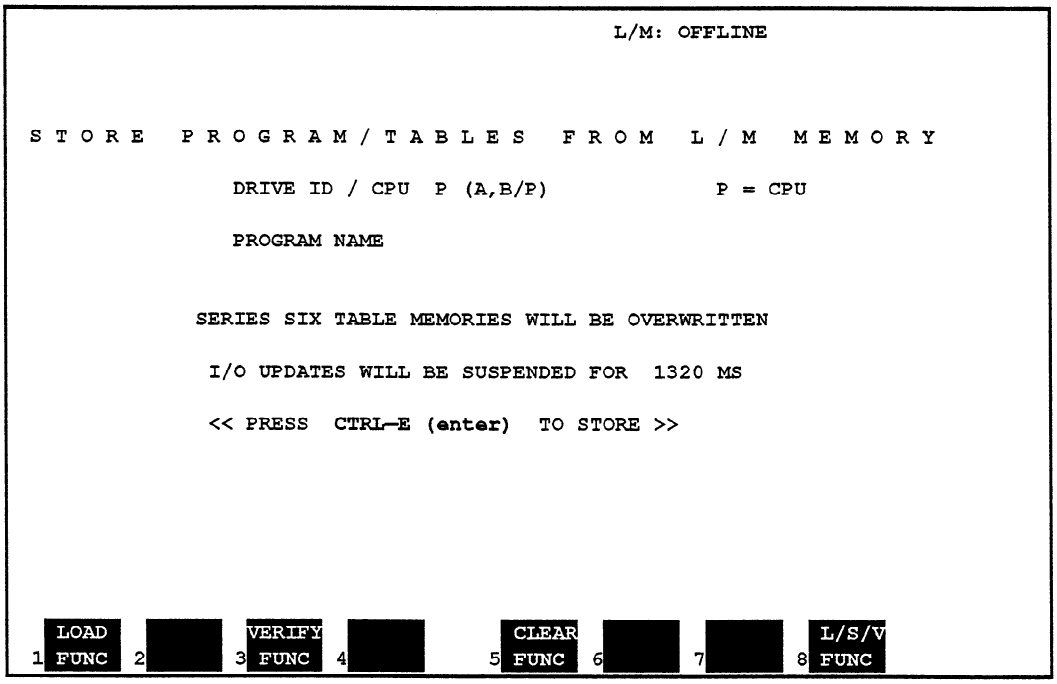
2. Move the cursor to "Program Name" and enter the name of the program.
3. Press CTRL-E (or the Enter key) to copy the program into RAM memory. The screen displays the word "BUSY" during the transfer.
4. If the system finds any fault or inconsistency in the data, the Load operation is terminated and an error message is displayed. After a successful load, the screen displays the message "LOAD COMPLETE."
5. To stop a program transfer in progress, press F4 (Abort).

Storing Data from RAM Memory

The Store function copies a program and its associated tables currently in system RAM memory. The transfer may be to a CPU, or to file storage on a hard disk or diskettes.

If the Logicmaster system is in Off-Line mode, the entire program, including the ladder logic and tables, is copied to the CPU. In On-Line mode, only the ladder logic is copied. If the system is in Monitor mode, the program cannot be copied into the CPU.

To use this function, select Store Function (F2) from the Load/Store/Verify menu.



NOTE

This menu is for the parallel version of the Logicmaster 6 software.

To store the program in RAM memory to a storage device, or to the CPU:

1. With the cursor at "Drive ID/CPU," enter the letter that represents the drive where the program will be stored, or the letter **P** for the CPU.

   To copy the program to the CPU, the Memory Protect switch on the CPU must be in the WRITE position and must remain in the WRITE position until the Store operation is completed.

2. Move the cursor to "Program Name" and enter the name of the program.

CAUTION

If a program with the same name is already stored on the disk, the new data will be written over it.

The CPU can only contain one program at a time. If a program is already stored in the CPU, the new data will be written over it and the original program in the CPU will be lost.

3. Press CTRL-E (or the Enter key) to copy the program. The system checks the size and functions used by the program. If the program is compatible with the CPU, the screen displays a message showing the number of seconds it must suspend a scanning CPU to copy the program.

NOTE

With the serial version of Logicmaster 6 software, the attached CPU must be stopped before the program can be transferred to the CPU. If the CPU is not stopped, the screen displays the message "SERIES SIX MUST BE STOPPED ."

4. While a program is being transferred to the CPU, the screen displays the message:

```
PROGRAM BEING TRANSFERRED TO THE CPU.
DO NOT PLACE CPU IN RUN OR WRITE DISABLED MODE.
```

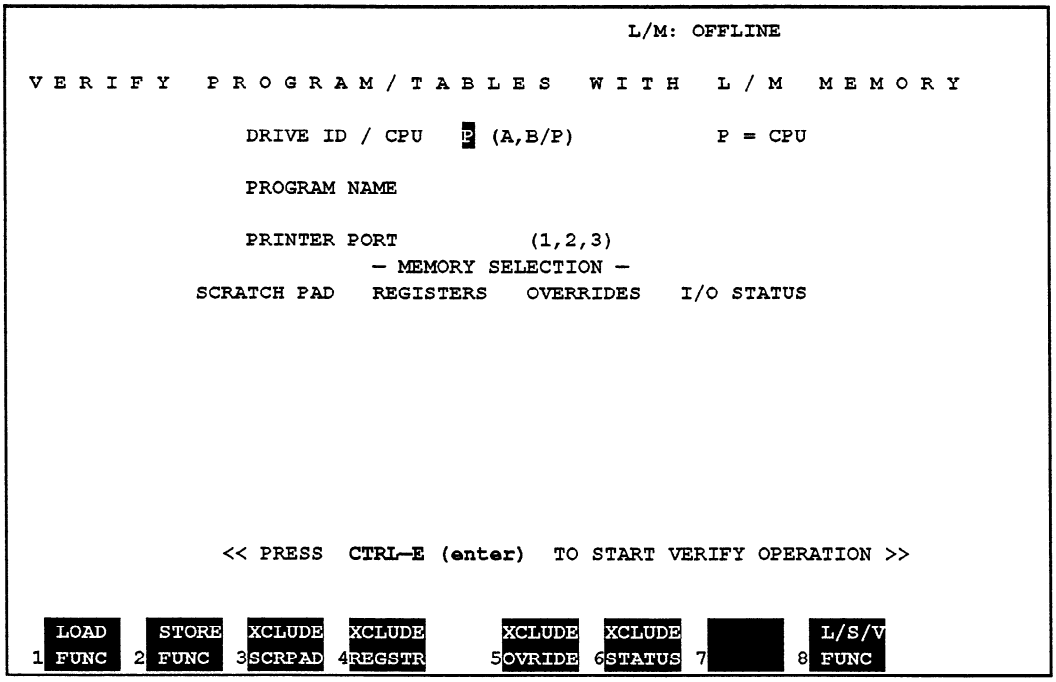
Placing the CPU in RUN mode during the Store operation with the serial version will result in the CPU containing an invalid program.

5. To stop a program transfer in progress, press F4 (Abort).

Verifying the Content of Program Data

The Verify function compares versions of a program after a Load or Store operation. Differences, called "miscompares", are displayed and can be printed out. If the verification is with a scanning CPU, dynamic data such as Scratch Pad, timer and counter registers, override table and I/O states can be omitted from the comparison.

To use this function, select Verify Function (F3) from the Load/Store/Verify menu.



NOTE

The Verify function compares relative memory locations. If one version of the program has an additional element, all the memory locations from that point on may miscompare.

To compare the program in RAM memory with the version in the CPU, or in storage:

1. With the cursor at "Drive ID/CPU," enter the character that represents the drive where the program to be compared is located. Enter the drive location of the program file, or the letter **P** for the CPU.
2. If the comparison is to a file on disk, move the cursor to "Program Name" and enter the name of the program.
3. To print a copy of the differences, move the cursor to "Printer Port" and enter the number of the port to which the printer is connected. Be sure that the printer is on-line and connected to the system.
4. To exclude tables from the comparison:
   A. Press F3 to exclude the contents of the Scratch Pad memory. The function of the F3 key then becomes Include Scratch Pad.
   B. Press F4 to exclude the contents of the registers. The function of the F4 key then becomes Include Registers.

C. Press F5 to exclude overrides. The function of the F5 key then becomes Include Overrides.

D. Press F6 to exclude the contents of I/O status tables. The function of the F6 key then becomes Include Status.

5. Press CTRL-E (or the Enter key) to begin the Verify operation. During the Verify operation, the screen displays the "BUSY" message. In addition, two key functions, Pause (F6) and Abort (F7), are displayed.

   A. To temporarily halt the Verify operation, press F6. To resume the operation, press F6 again.

   B. To end the operation before it is completed, press F7 and respond to the prompt to confirm the abort. This ends the function completely.

6. Any miscompares encountered are listed on the screen.

```
                    - MEMORY SELECTION -
SCRATCH PAD    REGISTERS    OVERRIDES    I/O STATUS

          MISCOMPARES (ALL DATA HEX EXCEPT SP)
LOCATION    W/M    OTHER       LOCATION    W/M    OTHER
SP    LM     16K      1K       SP    RM     8K      1K
SP    ID      1       20       SP    IS    xpd    ext
R  00055   0024     0000       R  00056   0025    000
R  00057   0026      000       PGM  0009H   F123    8700

        VERIFY COMPLETE  00008 MISCOMPARE(S) FOUND
```

This screen shows two columns of miscompares. Up to 10 miscompares can be displayed at once. If more than 10 miscompares are located, the 10 most recent are displayed. A printout has the same format as the screen display. Miscompares are listed from left to right by row:

```
(first)      (second)
(third)      (fourth)
(fifth)       etc.
```

Miscompare Screen: Definitions

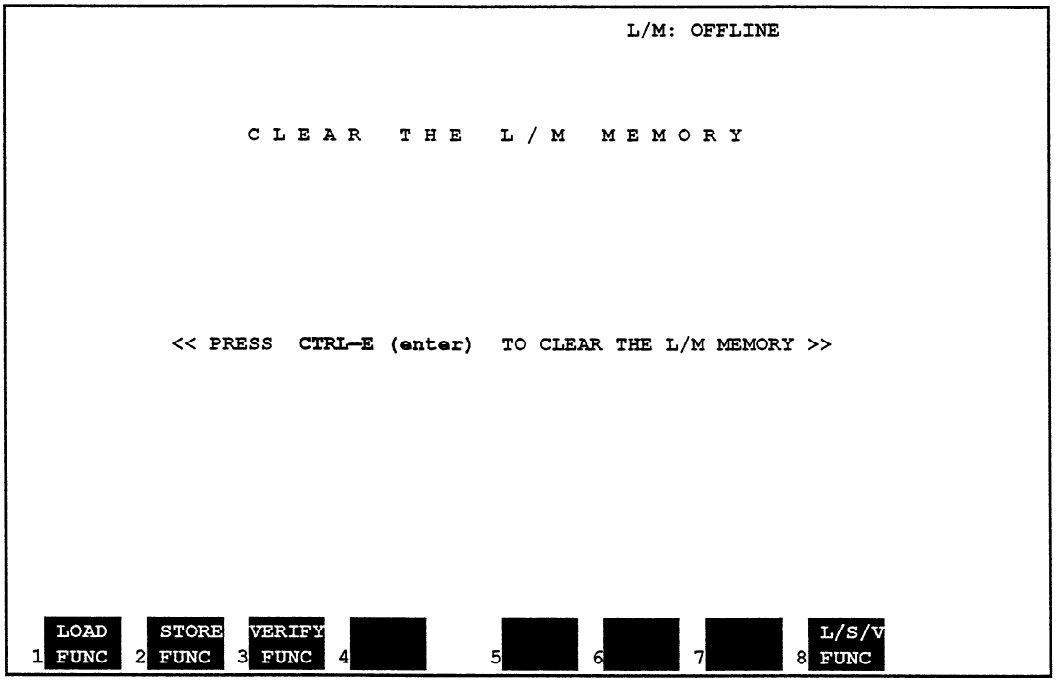
Each miscompare displays the memory type where the miscompare occurs, and gives the corresponding values in the program and in the CPU or storage. These items are abbreviated as shown below.

Abbreviation	Description
SP	Scratchpad memory.
OVR	Override memory.
O	Output status table.
I	Input status table.
IS	Instruction set (basic, advanced, extended, or expanded).
R	Storage registers.
PGM	Ladder program.
LM	Logic memory (size in K).
RM	Register memory (size in K).
ID	ID of the CPU.

Clearing RAM Memory

The Clear function deletes a program and its associated tables from the Logicmaster 6 system's RAM memory. This is usually done after storing the program to a drive, or to the CPU.

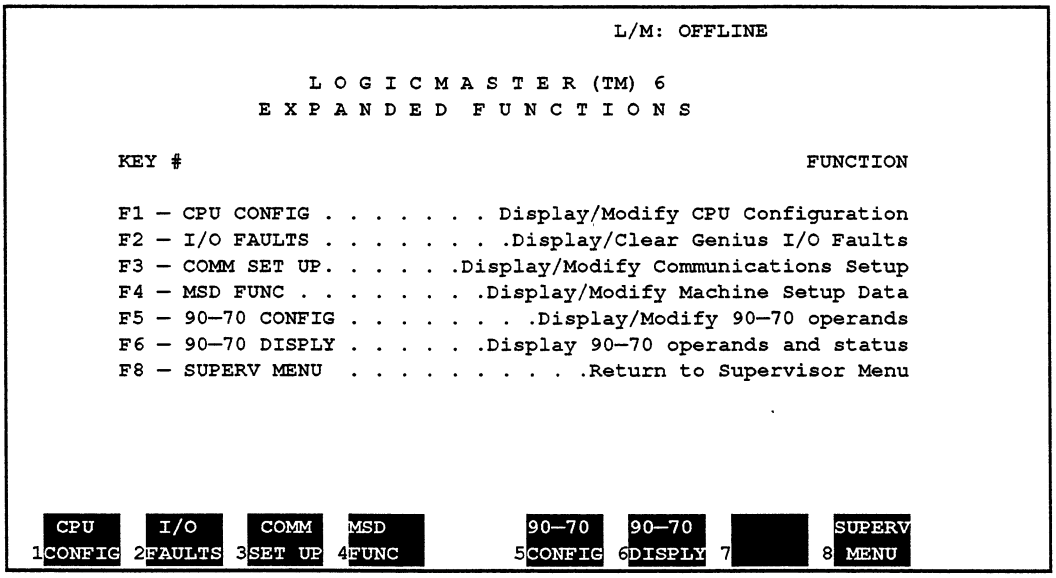
To use this function, select Clear Function (F5) from the Load/Store/Verify menu.



Press CTRL-E (or the Enter key) to delete the entire program from RAM memory. The screen will display the message "CLEAR COMPLETE."

Chapter 10
Expanded Functions Menu

The Expanded Functions menu provides screens to set up and display CPU configuration, Series 90-70 I/O configuration, Genius I/O fault reporting, and communications setup. To display the Expanded Functions menu, select Expanded Functions (F7) from the Supervisor menu.



Function Key	Function	Description
F1	CPU Configuration	Display the CPU Configuration menu, if the CPU Configuration instruction is present in the program.
F2	I/O Faults	Display the Genius I/O Faults screen, if the CPU Configuration instruction is present in the program.
F3	Communications Setup	Enter the parameters for communicating with the CCM card. This function is only available for serial versions of the software. For communication with the CPU, the CCM card must be at least a CCM2 with revision E firmware or a CCM3 with revision C firmware.
F4	MSD Functions	Create larger programs, or change programmer display colors.
F5	90-70 Configuration	Display or modify the communications parameters and output defaults of a Series 90-70 I/O rack, if the Service 90-70 I/O instruction is present in the program.
F6	90-70 Display	Display the communications parameters, output defaults, and rack status information for a Series 90-70 I/O rack, if the Service 90-70 I/O instruction is present in the program.
F8	Supervisor Menu	Return to the Supervisor menu.

Editing the CPU Configuration Setup Menu

The CPU Configuration Setup menu is used to enable Expanded I/O scan, Genius I/O diagnostics, and the Computer Mailbox. The Configuration Setup menu can only be displayed if the current program includes the CPU Configuration function. For more information, refer to chapter 14, *Expanded Functions*.

To display the Configuration Setup menu:

1. The program must be present in system memory. If it is not, load it using the Load/Store/Verify functions. Return to the Supervisor menu.
2. From the Supervisor menu, select Expanded Functions (F7).
3. From the Expanded Functions menu, press F1 (CPU Config).

```
                                                    L/M: OFFLINE

     C P U   C O N F I G U R A T I O N   S E T   U P   M E N U

EXPANDED I/O        ENABLED                  Y       (Y/N)
SCAN:
                    BEGIN RANGE    CHANNEL 0  POINT    1
                    END RANGE      CHANNEL 0  POINT 1024

GENIUS I/O          DIAGNOSTICS ENABLED      Y       (Y/N)
DIAGNOSTICS:        DIAGNOSTIC TABLES        Y       (Y/N)
                    B/C -> POINT FAULTS      N       (Y/N)
                    DIAGNOSTIC RANGE LIMIT 7         (0-7)   CHANNELS
                    CPU REGISTER SIZE  8,192                 REGISTERS
                    FAULT TABLE LENGTH       8               ENTRIES
                    BUS STATUS/CONTROL
                    BYTE LOCATION          993

COMPUTER MAIL BOX:  ENABLED                  N       (Y/N)




 1 B/C MAP  2        3        4        5        6        7        8 XPNDED FUNCS
```

Entries made on this screen change the program file in system memory and on disk when the function is exited; however, they do not change the program in the Series Six CPU. To change the values in the CPU, the program must be stored to the CPU using the Load/Store/Verify functions.

The entries on this screen are stored in the CPU Configuration function in the program. They can be printed out, as described in chapter 8, *Print Functions*.

To edit this display, move the cursor to the item you want to change, and type in the new value. Refer to the definitions that follow for more information.

To cancel entries and return to the Expanded Functions menu, press F9 (or the Abort key). The screen displays:

```
PRESS  ALT-X (CONFIRM)  TO ABORT, ANY OTHER KEY TO PROCEED
```

Press ALT-X (or the Confirm key) to return to the values that were displayed when the screen was entered. Press any other key to cancel the abort.

Expanded I/O Scan

Enabled: Enter **Y**, for Expanded I/O scanning by the CPU. For conventional I/O scanning, enter **N**.

Expanded I/O scan must only be enabled for the use of real world expanded channels and I/O devices. Expanded I/O is not enabled for internal use of expanded references.

When Expanded I/O is enabled, all real I/O points, including those in the main channel must be hardware-configured to be in a specific channel. This is done with a Genius bus controller for Genius blocks, or by an I/O Transmitter Module for Standard I/O. (Refer to GEK-96602, *Series Six Plus User's Manual*, for more information.)

Begin Range: I/O channels are scanned in pairs of one Main I/O chain channel and one Auxiliary I/O channel at the same time. If Expanded I/O scanning is enabled (above), this entry specifies the channel number pair and point number within the channels where the Expanded I/O scan should begin. The default values for Expanded I/O are Y = enabled. (0,1 = begin channel, point and 0,1024 = end channel, point). Change these entries to select a smaller scan range. Valid entries for Begin Range and End Range Channel are 0 to 7.

Each of these specifies a main/auxiliary chain channel pair:

Entry	Main Chain	Auxiliary Chain
0	I/O Status Table	Aux I/O Table
1	1	9
2	2	A
3	3	B
4	4	C
5	5	D
6	6	E
7	7	F

The value for point is the I/O point from 1 to 1024 in the specified channel. The system rounds this value to a byte boundary.

End Range: The channel number and point number within the channel where the Expanded I/O scan should end. This must be greater than the value for Begin Range.

Genius I/O

Diagnostics Enabled: Default = Y, to enable basic Genius I/O diagnostics. This creates a fault table in specified Series Six registers. The information includes fault type, I/O address of the faulted point or block, bus controller address, and time of the fault. This table is displayed by the Logicmaster software when I/O Faults (F2) is selected from the Expanded Functions menu.

For no Genius I/O diagnostics, enter **N**.

Diagnostic Tables: Default = N, to cause a Bus Controller (IOC) Status Bit Map to contain the locations of bus controllers that have reported errors. In addition, a Point Status Bit Map will specify any real I/O points that have been affected by a block, point, or bus controller fault. For no Genius I/O diagnostics tables, enter **N**. (Note: The diagnostic tables are not the same as the fault table.)

This function creates a bit map in the Series Six internal I/O, or minus (-) channels. Bits are set in the internal I/O addresses corresponding to any fault detected in the real I/O channels. The user must clear the bit in its logic. Faults include BUS CONTROLLER OK, ANY BLOCK FAULT, or POINT FAULT.

For example, a short circuit on a Genius block output point at O3+387 will cause O3-387 bit to be set to a 1. Even if the fault is cleared, O3-387 will remain on; it must be cleared by the user program. If this function is enabled, do not use any registers corresponding to the minus channels of Real I/O.

B/C - Point Faults: Default = N. This is meant as an extension of the diagnostic tables. For it to be used, bus controllers must be addressed on 256-bit boundaries, and all I/O blocks must be addressed following the bus controller. Its range only extends to the next available bus controller address, or 208 bits. (The bus controller itself takes the first 48 points.) If this function is enabled, the diagnostics monitors the bus controller status. It it fails, it will set all 208 bits in the input and output table in the minus (-) channel, following the bus controller status bytes.

For example, if a bus controller at address O1+0001 fails and B/C point faults is enabled:

```
O1-0049 to O1-0256:  All are set to 1.

I1-0049 to I1-0256:  All are set to 1.
```

Diagnostic Range Limit: The range by channel pairs of the diagnostics. Range = 0 to 7, as shown previously. Default = 7, which is 8 pairs of channels to be checked.

CPU Register Size: Enter the register size of the CPU:

256
1024 (can be represented by entering a 1)
8192 (can be represented by entering an 8)
16384 (can be represented by entering a 16)

Default = the number of registers indicated by the Scratch Pad.

The availablity of register memory regulates the number of I/O points on which Genius I/O diagnostics will be performed. If the register size is 256, only the first 256 inputs and outputs in the Main I/O chain are considered for diagnostics. For a register size of 1K, diagnostics are performed up to the first 1024 inputs and 1024 outputs of the Main I/O chain. Maximum register sizes of 8K or 16K allow the CPU to perform diagnostics on the maximum 16K inputs and 16K outputs.

The inputs and outputs included are the Main I/O chain (main I/O table, I/O channels 1+ to 7+) and the Auxiliary I/O chain (Auxiliary I/O table, I/O channels 9+ to F+).

Fault Table Length: The maximum size of the fault table depends on the entered value of the CPU register size and whether the Computer Mailbox has been enabled. Maximum table sizes are listed below. (Default = 8.)

	Maximum Table Length Computer Mailbox Enabled?	
Register Size	**No**	**Yes**
256	1	8
1k	75	68
8k	406	399
16k	1225	1218

To change the length of the fault table, enter a new value here. Note that each table entry is equal to 10 registers. Refer to chapter 12, *Programming*, for more information about the the size of the fault table in register memory.

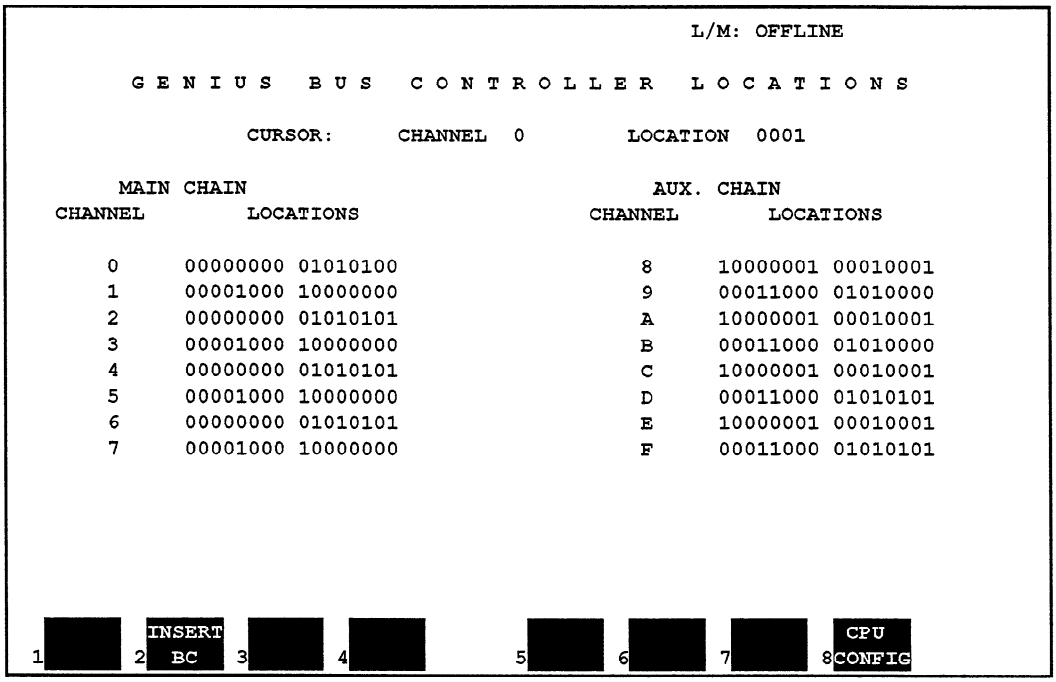
Bus Status/Control Byte Location: This is a group of 8 inputs and 8 outputs needed by the CPU diagnostics routine. It must be an unused address in the Main I/O table, but *not* the bus controller address.) This is the point address where the diagnostics discrete status (input table) and control (output table) bytes are located. Allowable entries are 0001 through 1017. (Default = 0993.)

Computer Mailbox: Default = N. To use the Computer Mailbox, change this entry to **Y**. If set to **Y**, the CPU will open a communications window to any valid address located in the first register of the computer mailbox. The window is opened once per sweep, only when the CPU is in Run mode. For more information about using the Computer Mailbox, refer to chapter 12, *Programming*.

Editing the Genius Bus Controller Locations Screen

The Genius Bus Controller Locations screen specifies the locations of the Genius I/O Bus Controllers within the Expanded I/O channels of the Series Six Plus PLC.

To display the Bus Controller screen, select Bus Controller Map (F1) from the CPU Configuration Setup menu.



The display shows the 16 Expanded I/O channels. The number 0 represents the Main I/O status table; the number 8 represents the Auxiliary I/O status table. Beside each number, states of 16 bits are represented. Current cursor position is shown at the top of the screen. The right-most bit in each channel represents I/O location 1. Each successive bit to the left is a location that is 64 greater. The leftmost bit in each channel represents location 961. Each bit with a value of 1 represents a bus controller location.

NOTE

The leftmost bit setting to a 1 results in an illegal bus controller location and should always be left set to 0.

To change the value of a bus controller bit setting, use the cursor keys or Return key to move the cursor to the bit. Then, press F2 (Insert BC) to change the bit.

To exit and restore the values that were displayed when the screen was entered, press F9 (or the Abort key). The screen prompts:

```
PRESS ALT-X (CONFIRM) TO ABORT, ANY OTHER KEY TO PROCEED
```

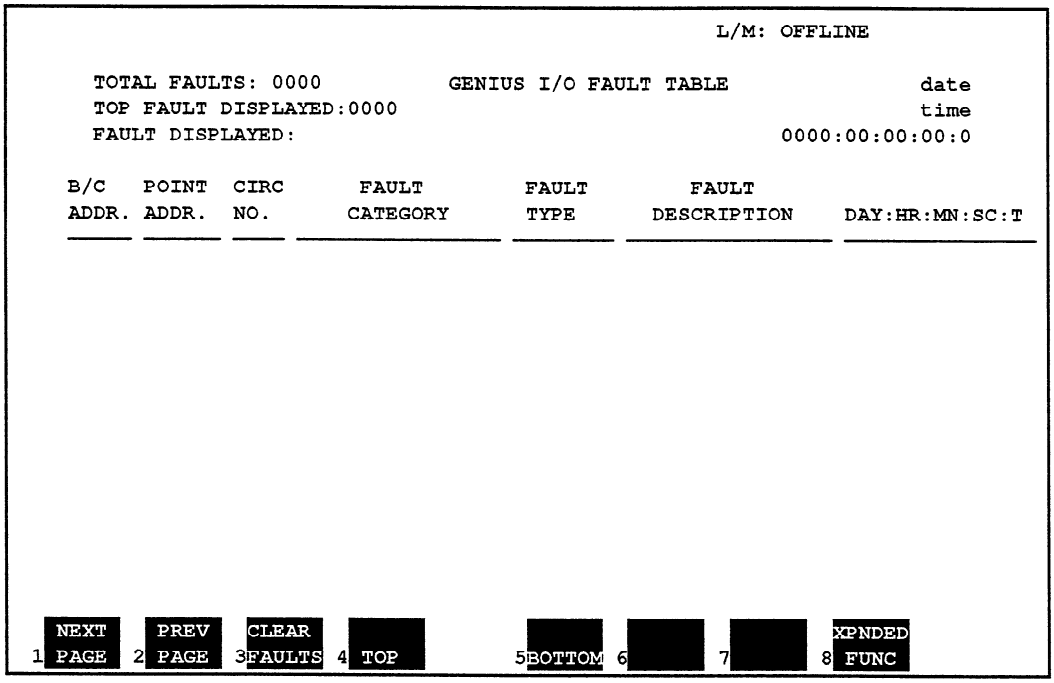
Press ALT-X (or the Confirm key) to exit, or press any other key to cancel the abort.

Clearing Genius I/O Faults

The Genius I/O Fault table display shows the Genius I/O faults. Faults are shown in order of occurrence, with the first fault at the top. The table can only be displayed if the current program includes the CPU Configuration function. In addition, Genius I/O diagnostics must be enabled in the CPU Configuration Setup menu, and the programmer must be in On-Line or Monitor mode.

To display the Genius I/O Fault table:

1. The program must be present in system memory. If it is not, load it using the Load/Store/Verify functions. Return to the Supervisor menu.
2. From the Supervisor menu, select Expanded Functions (F7).
3. From the Expanded Functions menu, press F2 (I/O Faults).



When the fault table is displayed on the screen, it can be printed using the ALT-P keys. (Printing must be set up as explained in chapter 11, *Utility Functions*).

If a fault occurs that has more than one description (see the definition of Fault Category, below), each description is listed as a separate line in the table. The entries in the table show the following information about a fault:

Bus Controller Address

The bus controller address is displayed for all faults. This entry has two fields, channel number and byte address.

1. **Channel Number:** The number of the channel where the error occurred, a hex value from 0 to F. The Main I/O status table is represented as 0; the Auxiliary I/O status table is represented as 8.

Main	Auxiliary
0	8
1	9
2	A
3	B
4	C
5	D
6	E
7	F

2. **Byte Address:** The byte boundary start address of the bus controller reporting the error. (Range = 0 to 125.)

Point Address

The point address is not displayed if the error is a bus controller or Serial Bus fault. This entry has two fields, input/output and address.

1. **Input/Output:** The first two characters indicate an input (I) or output (O). Both may appear for some faults.
2. **Address:** The address of the error. (Range = 1 to 1024.)

Circuit Number

The circuit number (relative circuit of the block) is displayed only for a circuit fault. (Range = 1 to 16.)

Fault Category

The fault category entry shows the category of error that has occurred.

IOC FAULT
BUS ERROR
CIRCUIT FAULT
LOSS OF BLOCK
BLOCK ADDITION
ADDRESS CONFLICT
EEPROM FAILURE
CALIBR MEM FAIL
SHARED RAM FAULT
INTL CIRCUIT FAULT

Fault Type

The fault type describes the error type: block, discrete, or analog. This is blank unless it is a circuit fault.

Fault Description

The fault description is displayed only if the fault category is "circuit fault." Multiple lines may be displayed if more than one description is associated with a fault data entry. Some example descriptions are:

EEPROM FAILURE
SHORT CIRCUIT
NO LOAD
SWITCH FAILED
HIGH ALARM
OVER RANGE
IOC NOT OK
LOSS OF MODULE
I/O ADDRESS CONFLICT
LOSS OF POWER
OVERLOAD
OVER TEMP
LOW ALARM
UNDER RANGE
OPEN WIRE
SERIAL BUS ERROR
ADDITION OF MODULE
IN WIRING ERROR
INTERNAL FAULT
INPUT CHAN SHORT

Fault Time

Fault time displays the value of the CPU's real-time clock when the error occurred. The time is displayed as day, hour, minute, second, and tenth of a second.

Viewing Additional Fault Listings

To display or print additional faults in the table:

1. To move the display down one line at a time, press the Next key or the Down Cursor key.
2. To move the display up one line at a time, press the Prev key or the Up Cursor key.
3. To move the display down one screen at a time, press Shift-Next or F1 (Next Page).
4. To move the display up one screen at a time, press Shift-Prev or F2 (Prev Page).
5. To go to the top of the table, press F4 (Top). To go to the end of the table, press F5 (Bottom).

Clearing Faults

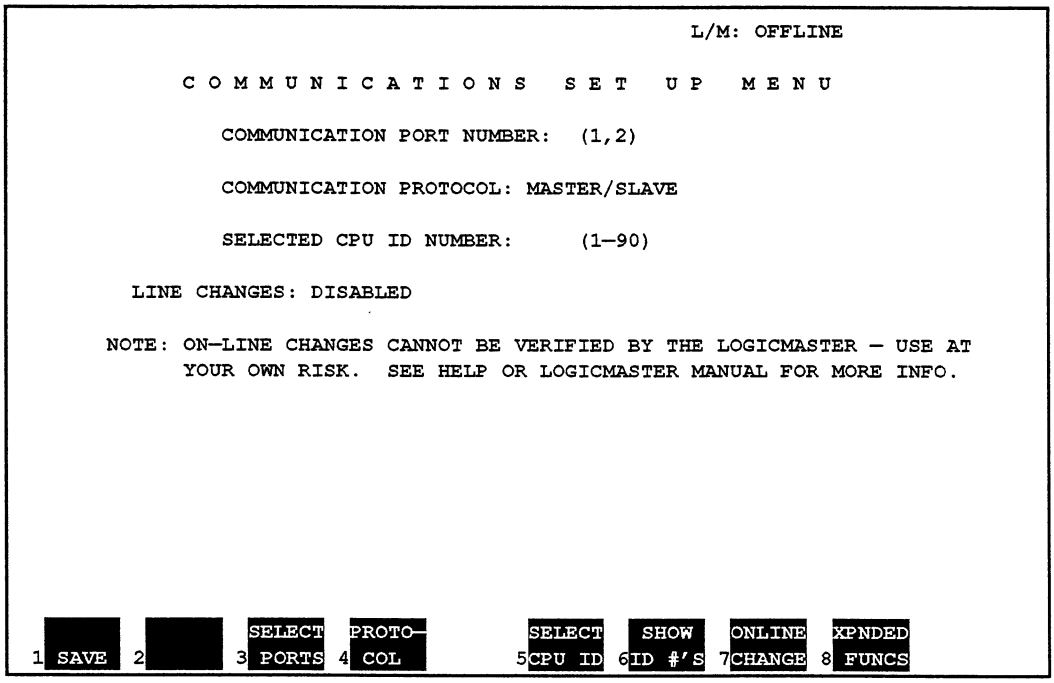
Press F3 (Clear Faults) key to clear the fault table. This sets the fault count to zero, and sends a Clear command to the Genius I/O blocks. Subsequent incoming faults fill the fault table from the beginning.

For the serial version of Logicmaster 6 software, on-line changes to the CPU must be enabled. For instructions, refer to section 4 of this chapter.

Setting up Communications with the CPU

The Communications Setup menu is used to specify parameters for serial communications with the CPU. This function applies only to the serial version of Logicmaster 6 software.

To display the Communications Setup menu, select Communications Setup (F3) from the Expanded Functions menu.



This screen shows the page format and function key assignments that are displayed for master/slave communication. At startup, the system uses data from the COMSET.SET file for this display. If the file is not present, default values are provided. That file can be changed by making new entries on this page and pressing F1 (Save).

Changes to the entries on the Communication Setup screen are made in two ways:

1. Port Number and CPU ID fields: Type the entry into the work area and press the appropriate function key.
2. Protocol and On-Line Changes fields: Press the appropriate function key to toggle the selections.

To exit this screen and return to the values that were displayed when the screen was entered, press F9 (or the Abort key). The screen displays:

```
PRESS  ALT-X (CONFIRM)  TO ABORT, ANY OTHER KEY TO PROCEED
```

Press ALT-X (or the Confirm key) to exit, or press any other key to cancel the abort.

Refer to the following definitions changing the Communication Setup screen:

Communication Port Number

This entry identifies the number of the CCM communication port. The menu displays the numbers of the ports that were operable when the system was started up.

To change the port number, enter in the work area one of the numbers that appears on the screen. To deselect a port, leave the work area blank. Press F6 (Select Port). Note that a port currently selected for communication must be deselected if it is to be used for another purpose.

NOTE

The port and protocol selected must be configured to match the CCM card(s) to which the system is connected, and the selected protocol. If the port is on the Combination Adapter card of the Workmaster computer, it can only be RS-232. Therefore, it can only be connected to a CCM card configured for RS-232. If the port is on the Asynchronous/Joystick card, it may be configured for either RS-232 or RS-422. If it is set up for RS-232, it can only be connected to a CCM card configured for RS-232. If it is set up for RS-422, it may be used with a CCM set up for RS-422.

Both master/slave and peer-to-peer protocols may be connected to either RS-232 or RS-422. Multidrop (one Logicmaster 6 system connected to more than one CCM) must be done with RS-422. Consult the *Data Communications Manual*, GFK-25364, for more information.

Communication Protocol

Press F4 (Protocol) to select either master/slave or peer-to-peer protocol. The protocol selection may not be changed from peer-to-peer to master/slave if the CPU ID is greater than 90. If master/slave is used, the Logicmaster 6 system will be the master and the CCMs must be configured to be slaves.

The system and the CCM must be set to the same protocol. Master/slave protocol is required for multi-drop operations (several CCMs in a common serial line). Either protocol can be used when one CCM is connected to the Logicmaster 6 system. Peer-to-peer communications require less system overhead.

Selected CPU ID

Enter the number of the CPU that will be communicating with the system. The number may be from 1 to 90 for master/slave communication, or 1 to 254 for peer-to-peer communication. Although 0 can be entered, it is not a valid ID for communication. ID 255 is used to broadcast to all CPUs, and can be used to initially establish communications if the actual CPU ID is not known.

After entering the number, press the Select CPU ID (F6) key. The number will be entered into the field, and the system will immediately begin communicating with that CPU.

To display the IDs of available CPUs within the parentheses to the right, press F6 (Show ID #s). For a network interface with more than eight CPUs, press the F6 key again, as needed, to display additional CPU IDs.

To cancel the polling of CPU IDs, press F9 (or the Abort key). The screen prompts:

```
PRESS ALT-X (CONFIRM) TO ABORT CPU POLLING,
ANY OTHER KEY TO PROCEED
```

Press ALT-X (or the Confirm key) to confirm the abort, or any other key to continue polling CPU IDs.

On-Line Changes

Press F7 (On-Line Change) to select the ability to use the system's on-line change features.

CAUTION

This feature should be used with care. Single-bit on-line changes (Toggle I/O) may inadvertently change bits other than the intended one.

When making single-bit changes, the system reads the value of the bit on one sweep of the CPU and writes back the command to make the change in a later sweep. If the bit changes after the read, that change will be overwritten by the CPU, resulting in an incorrect value.

In addition, the system does not verify the change until a later sweep. Therefore, if an error has occurred, the system cannot try the change again.

NOTE

If the CCM card in the CPU is earlier than version 5, and the CPU was programmed using SCREQ commands, the system may encounter write-protected memory. If so, the changes will not be permitted.

Save

Press F1 (Save) to create a file named COMSET.SET on the system startup diskette. The system diskette must be write-enabled to receive the file, and must be in the default drive. This file will store the current settings for the Communications Setup screen, except the listing of available CPU IDs. The system reads this file at startup.

Editing the Machine Setup Data

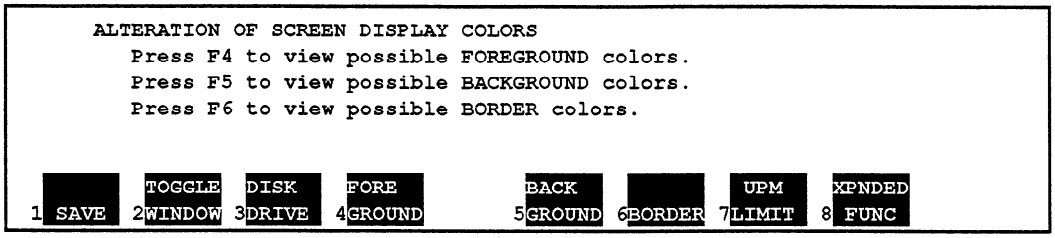
The Machine Setup Data (MSD) menu allows you to select program windowing or to change display colors on a color monitor. This data is stored in a file named MACHINE.SET, which is read at startup from system disk 1. On a hard disk, the file must be in the directory used to start up the Logicmaster 6 software. After you change selections on the MSD Setup menu, you must restart the computer to use the new data. Otherwise, the system will continue to use the data originally in the MACHINE.SET file. Before you restart the system, either by pressing the Restart button, by pressing the CTRL-ALT-DEL keys, or by cycling power to the computer, remember that:

- Any program file currently in memory will be lost. Save your program before restarting the computer.
- DOS must be present in the startup drive.

Changing Monitor Screen Colors

If you are using a color monitor, you can change the color of the text, background, and/or screen border.

1. Select MSD functions from the Expanded Functions menu. The lower part of the Machine Setup Data menu shows how to change display colors:



2. Use the function keys F4-F6 to toggle through the 16 different color selections for each item.
3. With system disk 1 in drive A, press F1 (Save) to save your entries.

Windowing

Larger programs can be built through the use of a file-handling feature called "Windowing":

	No Windowing	Windowing
Nicknames	2048	5096
Number of Rungs	6000	10000
Program Limit	32K	64K

Windowing works by temporarily storing part of the program on disk or RAM Disk, allowing programs to be created that are larger than the programmer's available memory. The disk selected for windowing should be the RAM Disk (Expanded Memory card) or hard disk in your computer, although a dual diskette drive computer can also use windowing. If the program will be very large and will include annotation, a hard disk should be used for windowing. *If a program is created using windowing, then windowing must always be selected for that program, as explained below.*

The .UPM (User Program Memory) File

At powerup, the computer reads the MACHINE.SET file on system disk 1 or in the directory used to start up the Logicmaster 6 software on a hard disk. If the file specifies that windowing is enabled, the Logicmaster 6 software creates another file named DISK.UPM on the drive selected for windowing. The DISK.UPM file, which takes up 32K bytes on the specified disk, is a working image of the program file. The DISK.UPM file must be present on the specified drive to load a .LAD file that was created with windowing enabled. If no DISK.UPM file is present on the disk, the program will not load, and an error message will appear. To load the program, it will be necessary to select windowing on the Machine Setup menu, and reboot the Logicmaster 6 software.

Loading a Program with more than 6000 Rungs

Windowing should be selected if a program that exceeds the 6000 rung non-windowing size will be loaded into the memory of the computer. Otherwise, the rungs over 6000 will be deleted from the program and from the disk if the program name is active. Therefore, before loading a program with more than 6000 rungs, be sure that windowing has been selected. If you attempt to load such a program when windowing is not selected in the MACHINE.SET file, a message will appear, asking whether you want to continue.

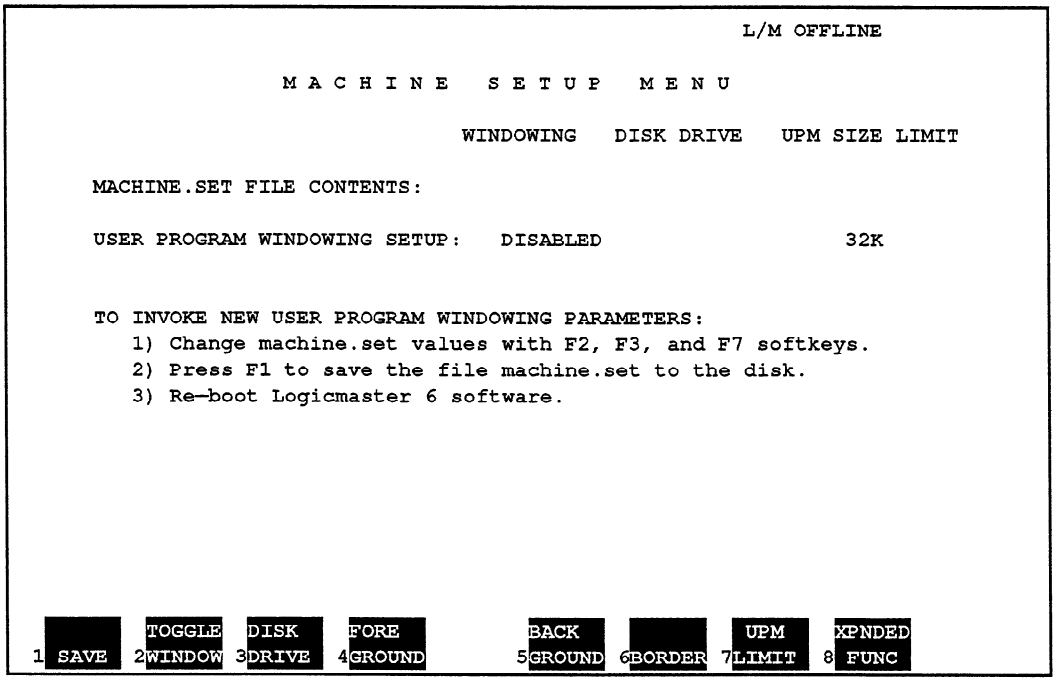
Editing a Program with more than 2K Nicknames

The total number of nicknames that can be used in a program is either 2K (2048) or 5K (5120), depending on whether windowing has been selected. Before editing nicknames in a program with more than 2K nicknames, be sure that windowing has been selected, as described on the previous page. Otherwise, the nicknames may not be stored correctly.

Selecting Windowing

To select or de-select windowing:

1. Select MSD Functions from the Expanded Functions menu. The upper part of the Machine Setup menu displays information about the defaults currently selected:



2. Press F2 (Toggle Window) to select either **Y** or **N** for windowing.
3. Place system disk 1 in drive A.
4. Press F3 (Disk Drive) to select the drive to be used for windowing. Each time F3 is pressed, a drive designation appears on the screen. Enter the correct drive letter. For example, for a RAM Disk designated as drive C, you would select C using the F3 key. If the computer has two diskette drives only, drive B is usually selected for windowing.

5. To create a program that is larger than 32K, press F7 (UPM Limit). The UPM (User Program Memory) size 64K appears on the screen.
6. You must press F1 (Save) to save your entries.
7. Using DOS, reboot the system using system disk 1. This is necessary to use the entries you have just made.

Editing the Series 90-70 I/O Setup

The Series 90-70 I/O Rack Configuration screen makes it easy to enter the configuration of each Series 90-70 I/O rack in the system. Before you enter the rack configuration on this screen, be sure the program currently in Logicmaster 6 system memory includes the 90-70 service instruction for that rack. For information about using that instruction in a program, refer to chapter 14, *Expanded Functions*.

To display the Configuration Setup menu:

1. The program including the 90-70 service instruction must be present in Logicmaster 6 system memory. If it is not, load it using the Load/Store/Verify functions. Then, return to the Supervisor menu.
2. From the Supervisor menu, select Expanded Functions (F7).
3. Press F5 to display the Series 90-70 I/O Rack Configuration screen.

```
                                                L/M OFFLINE

        9 0 – 7 0   1 / 0   R A C K   C O N F I G U R A T I O N

        RACK NUMBER:  04     RACK ADDESS RANGE: I/O    001–0256

                    RACK STATUS LOCATION:R8087 – R8089

            << PRESS ACCEPT  TO STORE RACK CONFIGURATION >>

         COMMUNICATIONS PARAMETERS              OUTPUT RESET ACTION

        PARAMETER 1 2 3 4 5 6 7 8          SLOT NUMBER    2 3 4 5 6 7 8 9
                  – – n – n – n –          RESET ACTION   0 0 0 0 0 0 0 0

        1. not used                                        0 = turn off
        2. not used                                        1 = hold last state
        3. enable comm timeout
        4. not used
        5. ignore CPU reset
        6. not used
        7. block downstream i/o faults
        8. not used

  COMM      RESET                                  GO TO     PREV      NEXT      EXP
1 TOGGLE  2 TOGGLE  3           4                5 RACK    6 RACK    7 RACK    8 MENU
```

This screen displays the following information about the lowest-numbered I/O rack entered in a Series 90-70 I/O service instruction:

Rack Number: This number (0-15) identifies the I/O rack.

Rack Address Range: I/O Status table locations for the I/O in the rack.

Rack Status Location: The location in CPU memory where status information from the rack will stored.

This screen also shows the current communications parameters and output defaults for the rack. These can be changed, as described on the following pages. It is important to note that changes on this screen will change the program file in system memory, and on disk when CTRL-A (or the Accept key) is pressed. They do not change the program in the Series Six CPU. If a file name is active and the system is in On-Line or Monitor mode, only the ladder logic portion of the .LAD file is updated when exiting the I/O Rack Configuration menu. If the system is in Off-Line mode, both the program and the tables are updated. If a file name is active but no corresponding .LAD file existed upon entering the CPU Configuration menu, pressing CTRL-A (or the Accept key) writes both the program and tables to a new .LAD file.

Setting Communications Parameters

The I/O rack may use the following communications parameters:

Enable Comm Timeout: determines the state all outputs in the rack will assume if CPU commands to the rack are suspended for more than 1 second. If **Y** is selected, all outputs are forced to an OFF state. If **N** is selected, all outputs hold the last state until CPU commands are resumed, or until a reset condition occurs in the rack.

Ignore CPU Reset: determines the state of all outputs in the rack if a Reset signal is received from the CPU. The CPU issues a Reset signal whenever an I/O parity error or I/O chain fault occurs. If **Y** is selected, all outputs are set to their default state, selected on this screen. If **N** is selected, all outputs are forced to an OFF state.

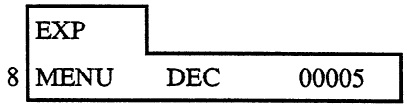
Block Downstream I/O Faults: determines whether certain types of fault reports from downstream racks will be transmitted to the CPU. If **Y** is selected, certain downstream faults are blocked, allowing the CPU to continue running, and servicing racks upstream of the fault. If **N** is selected, all downstream faults are transmitted to the CPU. If such a fault occurs, the CPU will stop operating.

The current selections for the rack are shown in the left center of the screen:

```
COMMUNICATIONS PARAMETERS

PARAMETER  1 2 3 4 5 6 7 8
           - - n - n - n -
```

At first, each of these parameters is set to **n** for not active. If you want to change any of the communications parameter selections for the current rack:

1. Select the parameter you want to change.
   A. Press CTRL-S (or the Select key) to move the work area cursor to the bottom line.
   B. Type in the number of the parameter you want to change. For example, if you want to change the Ignore Communication Timeout parameter, type **5**. The work area shows:



2. Press F1 (Comm Toggle) to change the current selection for that parameter. The parameter line changes to show the new selection. For example:

```
PARAMETER  1 2 3 4 5 6 7 8
           - - n - y - n -
```

Setting the Output Reset Action

The output reset actions for each output board in the rack can be selected individually. This will determine the state of all outputs on the board following a reset or loss of CPU communications.

Current output defaults for the rack are shown in the right center of the screen:

```
OUTPUT RESET ACTION

SLOT NUMBER     2 3 4 5 6 7 8 9
RESET ACTION    0 0 0 0 0 0 0 0
```

At first, the reset action for each board in the rack is set to 0. In this state, all outputs on the board are cleared (set low) following a reset or CPU communication timeout. If the selection for reset action is set to 1, outputs are latched to their last state. If you want to change the reset action for any module in the current rack, follow these steps:

1. Select the slot number you want to change (slot 1 is reserved for the Series Six I/O Receiver Module). With work area cursor to the bottom position, type in the number of the slot. For example, if you want to change slot 7, the work area will show:

```
  EXP
8 MENU    DEC      00007
```

2. Press F2 (Reset Toggle) to change the selection for that slot. The screen changes to show the new selection. For example:

```
 SLOT NUMBER 2 3 4 5 6 7 8 9
RESET ACTION 0 0 0 0 0 1 0 0
```

Configuring Additional I/O Racks

Complete the Series 90-70 I/O Rack Configuration screen for every rack for which a Series 90-70 I/O Rack service instruction has been included in the ladder logic program. Press F5 (Go To Rack), F6 (Previous Rack), or F7 (Next Rack) to display another screen.

When the entries on the screen are correct, press CTRL-A (or the Accept key) to store the new rack configuration to memory and to the .LAD file. Note that this will not change the program in the CPU. To do that, you must use the Store function, as explained in chapter 9, *Load/Store/Verify*. Press F8 to return to the Expanded Functions menu.

Displaying the Series 90-70 I/O Rack Display Screen

The Series 90-70 I/O Rack Display screen shows the current configuration of each rack in the system. To display this screen, the current program must include the 90-70 service instruction for that rack. For information about using that instruction in a program, refer to chapter 14, *Expanded Functions.*

To display the Series 90-70 I/O Rack Display screen:

1. The program including the 90-70 Service instruction must be present in Logicmaster 6 system memory. If it is not, load it using the Load/Store/Verify functions. Then, return to the Supervisor menu.
2. From the Supervisor menu, select Expanded Functions (F7).
3. From the Expanded Functions menu, press F6 (90-70 Display).

```
                                                    L/M: ONLINE

              9 0 - 7 0   1 / 0   R A C K   D I S P L A Y

RACK NUMBER:  04                  RACK ADDESS RANGE: I/O    001-0256
RACK STATUS LOCATION:R8087 - R8089

                    SLOT NUMBER:           2 3 4 5 6 7 8 9
                    INPUT MODULE ACTIVE:   n n n n n n n n
                    OUTPUT MODULE ACTIVE:  n n n n n n n n
                    OUTPUT RESET ACTION:   0 0 0 0 0 1 0 0
                    MODULE PRESENT:        n n n n n n n n

    COMMS PRMTRS: 1 2 3 4 5 6 7 8          RACK STATUS: 1 2 3 4 5 6 7 8
                  - - n - n - n -                       n n n - n - I -
    1. not used                            1. i/o module failure
    2. not used                            2. i/o chain fault
    3. enable comm timeout                 3. power supply fault
    4. not used                            4. not used
    5. ignore CPU reset                    5. communication timeout
    6. not used                            6. not used
    7. block downstream i/o faults         7. rack position
    8. not used                            8. not used

1 [      ] 2 [      ] 3 [      ] 4 [      ]  5 GO TO RACK  6 PREV RACK  7 NEXT RACK  8 EXP MENU
```

When the screen appears, it shows the following information about the lowest-numbered I/O rack in the system:

Rack Number: the number (0-15) of the I/O rack being configured.

Rack Address Range: the I/O Status table locations used by the I/O in that rack.

Rack Status Location: the location in CPU memory where status information from the rack is stored.

Communication Status: shows whether the module is currently communicating with the CPU.

Configuration of Input and Output Modules

The center of the screen shows the configuration of all slots in the rack:

SLOT NUMBER:	2	3	4	5	6	7	8	9
INPUT MODULE ACTIVE:	n	n	n	y	y	y	n	n
OUTPUT MODULE ACTIVE:	y	y	y	n	n	n	n	n
OUTPUT RESET ACTION:	1	0	1	0	0	0	0	0
MODULE PRESENT:	y	y	y	y	y	y	y	n

Slot 1 in the rack is reserved for the Series Six I/O Receiver Module. For each slot from 2 to 9, the display shows whether there is an I/O board in that slot. The display also shows which boards are currently active. In the illustration above, there is a board indicated in slot 8 of the rack, but no input or output board is shown as being currently active.

For output boards, the selected reset action is also shown (1=latched, 0=cleared). In the illustration, for example, there are output boards in slots 2, 3, and 4. All outputs on the boards in slots 2 and 4 default to all outputs latched to their last state following a reset. The board in slot 3 defaults all outputs to cleared after reset.

Rack Communications Parameters

The left center of the screen shows the communications parameters used by the rack.

```
COMMS PRMTRS:  1 2 3 4 5 6 7 8
               - - n - n - n -
```

The letters **n** and **y** show which of these three parameters are used:

Number	Parameter	Description
3	Enable Communications Timeout	Shows how all outputs in the rack will respond if CPU commands to the rack are suspended for more than 1 second. If **y** is selected, all outputs will be forced OFF. If **n** is selected, all outputs will hold their last state until CPU commands are resumed, or until a reset condition occurs in the rack.
5	Ignore CPU Reset	Shows how all outputs in the rack will respond if an I/O parity error or I/O chain fault occurs. If **y** is selected, all outputs will be set to their default state, selected on this screen. If **n** is selected, all outputs will be forced OFF.
7	Block Downstream I/O Faults	Shows whether certain types of fault reports from downstream racks will be transmitted to the CPU. If **y** is selected, certain downstream faults will be blocked, allowing the CPU to continue running, and servicing racks upstream of the fault. If **n** is selected, all downstream faults will be transmitted to the CPU. If a fault occurs, the CPU will stop operating.

Rack Status Parameters

The right center of the screen shows current status of the rack.

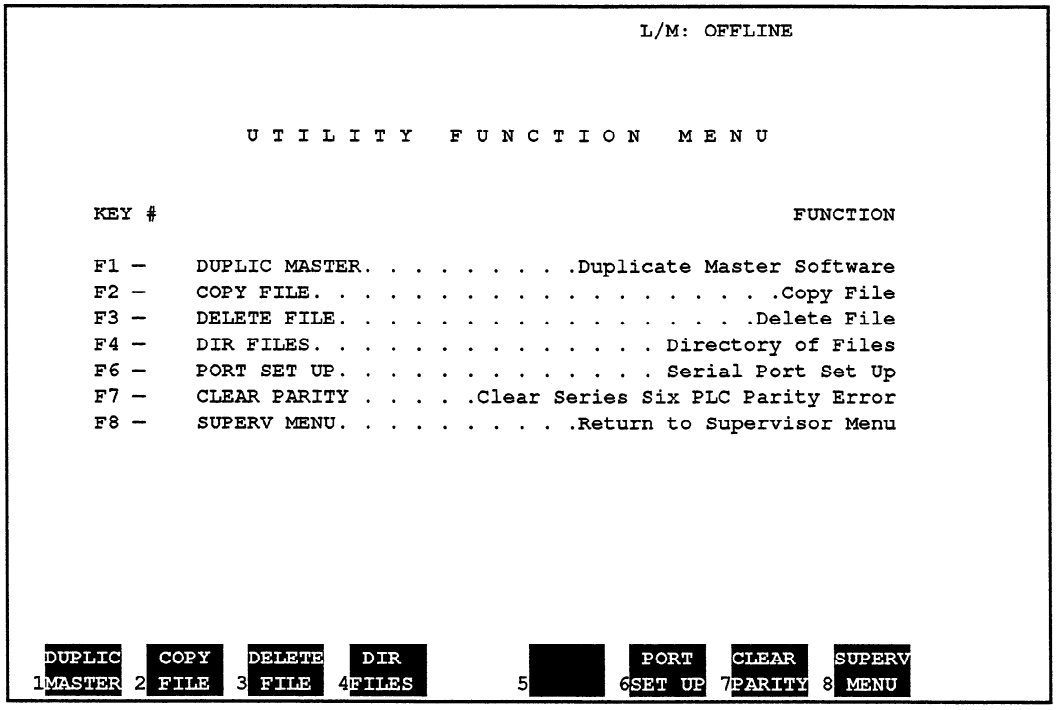
```
RACK STATUS:  1 2 3 4 5 6 7 8
              n n n - n - I -
```

The letters show the current status of each of the following:

Number	Parameter	Description
1	I/O Module Failure	If the status is y, a fault has been detected in one of the I/O boards in the rack.
2	I/O Chain Fault	If the status is y, a fault has been detected in a downstream rack.
3	Power Supply Fault	If the status is y, a failure has been detected in the rack power supply.
5	Communication Timeout	If the status is y, the board has not received communications from the CPU for more than 1 second. (For an output board, this defaults all outputs.)
7	Rack Position	Shows whether the rack is an intermediate (I) rack, or the last (L) rack on the bus.

Chapter 11
Utility Functions

Utility functions are used for disk and file management, for configuring the serial port(s), and for clearing parity errors in the CPU. To display the Utility Function menu, select Utilities (F8) from the Supervisor menu.



Function Key	Function	Description
F1	Duplicate Master	Make copies of the master diskette.
F2	Copy File	Copy one or more files.
F3	Delete File	Delete one or more files.
F4	Directory	Display a listing of the files on a diskette. This command will also print out the file listing if a printer is set up, and is on-line to the system.
F6	Port Setup	Specify the parameters of the serial port(s).
F7	Clear Parity	Remove parity errors in the Scratch Pad and Transition tables in the CPU memory.

Duplicating the Master Software

Use the Duplicate Master Software utility to copy the Logicmaster 6 master diskettes.

The Duplicate Master Software utility installs the Logicmaster 6 software on the hard disk. For complete installation instructions, turn to chapter 2, *Operation.*

On a diskette system, use the Dpulicate Master Software utility to make a copy of each master software diskette.

1. Format the diskette using DOS. For instructions on formatting diskettes and on using DOS, refer to chapter 2, *Operation.*
2. Use the Duplicate Master utility to copy the Logicmaster 6 system files onto the formatted diskette. Note that copies can only be made from the master diskette. Copy diskettes cannot themselves be copied.
3. Instructions are provided below for using the Duplicate Master Software utility with single and multiple floppy-diskette drives. Continue at the appropriate heading.

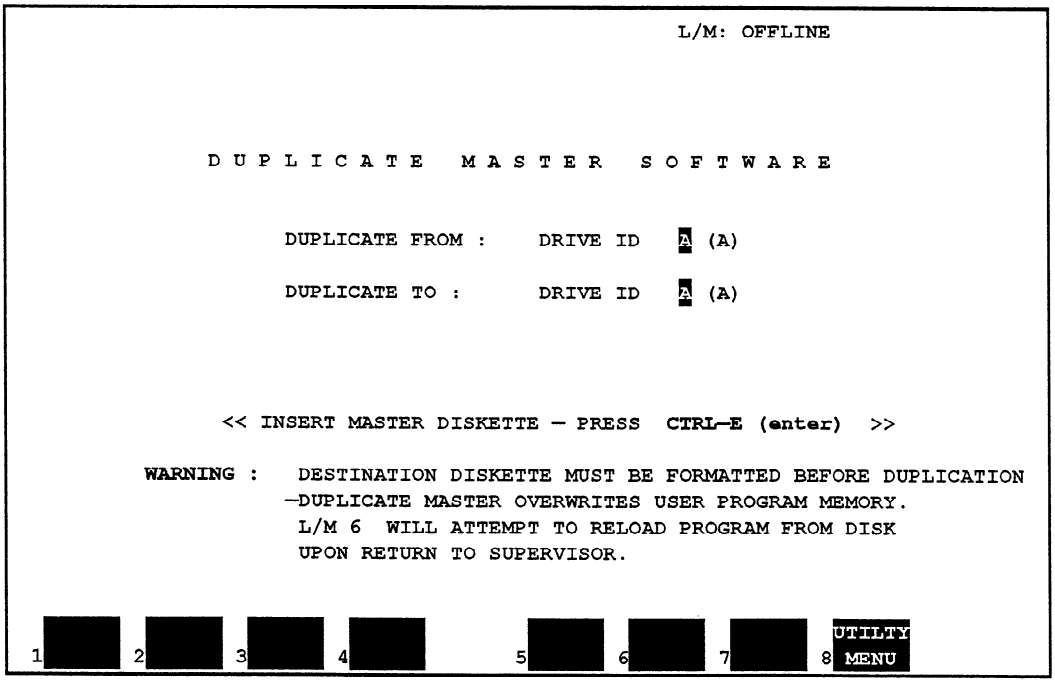
NOTE

If there is currently a program stored in the system's RAM memory, the Duplicate Master Software utility will destroy it. (If there is a program in RAM memory that you want to keep, be sure that you copy the program using the Load/Store/Verify function before you use the Duplicate Master Software utility.) The system will attempt to reload the program from disk when you return to the Supervisor menu.

Single Diskette-Drive System

To install Logicmaster 6 software on a single-drive system:

1. To display the Duplicate Master Software screen, press F1 (Duplicate Master) from the Utilities menu.



2. Place the master diskette (not a copy) in the drive.
3. Press CTRL-E (or the Enter key). When prompted, remove the master diskette.
4. Insert a formatted diskette in the drive. Press CTRL-E (or the Enter key).
5. After some of the program has been placed on the diskette, a prompt will appear. In response to the prompt, remove the copy (destination disk) and reinsert the master diskette.
6. Continue exchanging diskettes until the entire program has been copied. When the copying is complete, the following message appears at the bottom of the screen:

```
DUPLICATION COMPLETED
```

The next return to the Supervisor level causes the system to search for the program file that was used last, and to try to reload it if there is an active file name.

Multiple Diskette-Drive System

To install Logicmaster 6 software on a multiple-drive system:

1. To display the Duplicate Master Software screen, select Duplicate Master (F1) from the Utilities menu.

```
                                                    L/M: OFFLINE


          D U P L I C A T E   M A S T E R   S O F T W A R E

               DUPLICATE FROM :    DRIVE ID   A (A,B)

               DUPLICATE TO :      DRIVE ID   A (A,B)


            << INSERT MASTER DISKETTE - PRESS  CTRL-E (enter) >>

       WARNING :   DESTINATION DISKETTE MUST BE FORMATTED BEFORE DUPLICATION
                   -DUPLICATE MASTER OVERWRITES USER PROGRAM MEMORY.
                    L/M 6  WILL ATTEMPT TO RELOAD PROGRAM FROM DISK
                    UPON RETURN TO SUPERVISOR.

 1        2        3        4        5        6        7        8 UTILTY
                                                                  MENU
```

2. If the master software diskette (not a copy) is not presently installed, place it into a drive. Insert a formatted diskette in another drive (usually drive B).
3. At DUPLICATE FROM, enter the designation of the drive containing the master diskette.
4. Move the cursor to DUPLICATE TO and enter the designation of the drive to receive the copy.

5. Press CTRL-E (or the Enter key). The system begins copying the software onto the diskette. When the copy is complete, the message DUPLICATION COMPLETED appears. The next return to the Supervisor level causes the Logicmaster 6 system to search for the program file used last, and to attempt to reload it if there is an active file name.

Using File Utilities

The system stores all program data as files. Each file has a unique name by which the system identifies it. The name you give a program becomes the basic file name for that program, and for other files associated with it.

Do not begin a file name with any of the following: CON, AUX, COM1, COM2, PRN, LPT1, LPT2, LPT3, or NUL. These reserved file names have special meaning to the system. Do not use the wildcard characters * or ? as part of a file name.

''Wildcard'' characters can be used to represent parts of existing file names. When using wildcard characters in a file name, you must also use the period character between the main part of the file name and the extension (for example, PROG*.*). The two wildcard characters are:

*	The asterisk can be used to represent one or more characters in a file name. For example, an asterisk may represent all files with the same extension (*.LAD) or all extensions to the same file name (PROGRAM1.*).
?	The question mark can be used to represent one character in a file name (i.e., PROGRAM?.LAD or PROGRAM1.L??).

Both wildcard characters can be used together. For example, PROGRAM?.* means all files beginning with PROGRAM.

Program Files

A program is stored as more than one type of file. The system automatically gives each type of file a three-character file name extension. The extension is part of the file name, and differentiates one program file from another. When entering a file name and an extension, be sure to include the period between the file name and the extension.

The following file name extensions are used by the system:

Extension	Description
name.LAD	The .LAD file contains the ladder diagram, properly formatted for transfer to the CPU. This file uses two bytes of storage for each word of logic. The .LAD file includes registers, I/O status, overrides, and other information associated with the ladder diagram.
name.LBU	The .LBU file contains a copy of the .LAD file. The .LBU file is generated when program backup is selected at the beginning of the Edit Program function. If program backup is not selected, the .LBU file is not created.
name.RDF	The .RDF file contains the formats that will be used for reference displays and printouts. Initially, this file contains default formats supplied by the system. The Display References utility changes this file.

Extension	Description
name.RBU	The .RBU file is the a file for the .RDF file.
name.TXT	The .TXT file is the text output file that is generated when the program is stored for later printing, using the Print Program function.
name.NAM	The .NAM file stores program names and nicknames.
name.EXP	The .EXP file stores program rung explanations and coil labels.
name.NBU	The .NBU file is a backup file for the .NAM file.
name.EBU	The .EBU file is a backup file for the .EXP file.

If you had a program named PROGRAM1, the ladder diagram would be stored as a file named PROGRAM1.LAD. Other files for that program would be named as described above. For example, the format for the reference displays would be named PROGRAM1.RDF.

If you have programs created with Release 2.02 or earlier of the Logicmaster 6 software, the first time you Edit, Display, or Print them with Release 3.01 or later software, Logicmaster 6 converts the structure of the .NAM, .EXP, and .RDF files. That makes these files incompatible with earlier versions of the software. You should first copy these files onto a newly-formatted program diskette, and use these copies. If a disk error occurs while the files are being converted for editing, copy only the .NAM, .EXP, and .RDF files from the diskette, and try again.

In addition to the program files, other files include:

Extension	Description
COMSET.SET	The Communications Setup file stores the communications setup data for serial versions of the software. The system creates or updates this file when the Save (F1) key is pressed from the Communications Setup menu.
DISK.UPM	The system creates the User Program Memory file when program windowing is selected. This file must be stored on the program disk.
FX.DEF	The Function Definitions file stores the special function key assignments created for the Logicmaster 6 system.
MACHINE.SET	The Machine Setup file sets up program windowing and color monitor default colors. This file must be stored on system disk 1. If a hard disk is used to start up the Logicmaster 6 software, this file must be present in the startup directory. The system reads this file when the Logicmaster 6 software is started up. If the machine setup is changed, the system must be restarted with the new MACHINE.SET file present to use the new selections.
PORTn.PSU	The PORT1.PSU or PORT2.PSU file contains the serial port setup information entered with the Setup Serial Port utility.
PRINTER.SET	The Printer Setup file stores the printer data. The system creates or updates this file when the Save (F5) key is pressed from the Define Printer screen.

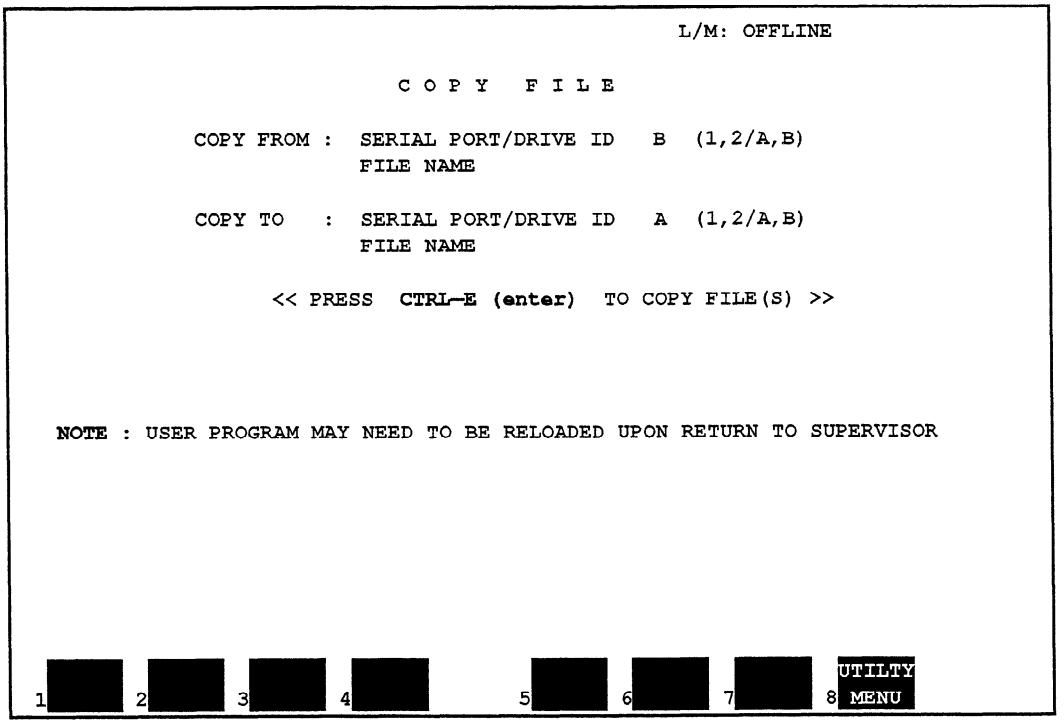
Copying Files

The Copy File utility is used to copy one or more files. Copying may be done between serial ports or disks.

NOTE

When the Copy File utility operates, it temporarily uses the system's RAM memory to store the files that are being copied. If there is a program in RAM, it may need to be reloaded after using the Copy File utility. The system will reload the program file automatically upon returning to Supervisor level, if the program file is available (and the file name is active).

To display the Copy File screen, press F2 (Copy File) from the Utilities menu.



1. Enter the designation of the serial port or drive from which the file(s) will be copied.
2. Enter the name of one or more files to be copied. Use wildcard characters as needed in the file name. For example, enter *.LAD to copy all ladder diagram files on the source drive, or (name).* to copy all files with the same program name. For more information on wildcards, refer to ''Using Wildcards to Represent Parts of a File names''.

   To copy the primary program files (.LAD, .RDF, .NAM, and .EXP files), enter just the file name without an extension.
3. Enter the designation of the serial port or drive to receive the file(s).
4. If the copies are to have the same name as the original versions of the files, no entry is needed for the COPY TO file name.

Enter a new name if you want to rename the copies. To name multiple file copies, use wildcard characters as needed. For example, you could enter the file name PROGRAM1.* to copy all the PROGRAM1 files, and enter PROGRAM2.* as the file name for the copies.

NOTE

When entering the name for the copy, be sure not to use the name (including the file name extension) of a file already stored on the destination device. The new file will replace any old file with the same name, and the old file will be lost.

5. Press CTRL-E (or the Enter key). If the entries for source and destination are correct, the copy begins.
6. Respond to any prompts to insert and remove diskettes. When the message:

```
File Copy Completed
```

appears, the copy is finished. If multiple files were copied using wildcard characters, the names of the files will appear on the screen.

Renaming Backup Files

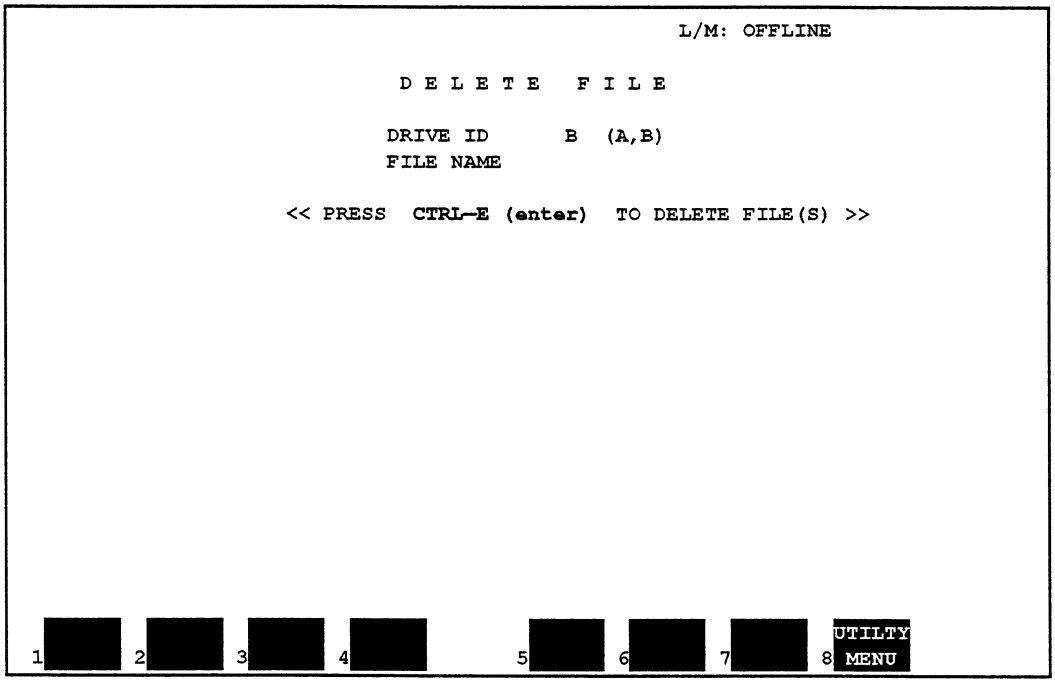
Backup files must be renamed to be accessed and used as program files.

Type of File	Program File Name	Backup File Name
Ladder logic	program.LAD	program.LBU
Rung explanations and coil labels	program.EXP	program.EBU
Names and nicknames	program.NAM	program.NBU
Reference display formats	program.RDF	program.RBU

Use the Copy utility to copy a backup file, giving the copy a new name and the appropriate extension, as listed above. For example, program1.LBU could be renamed program2.LAD. This new file could be edited and backed up like any other program file.

Deleting Files

The Delete File utility is used to delete one or more files from a disk. To display the Delete File screen, press F3 (Delete File) from the Utilities menu.



1. Enter the designation of the drive from which the file(s) will be deleted.
2. Enter the name of one or more files to be deleted. Use wildcard characters as needed to specify multiple files. For example, enter *.TXT to delete all printer text files on the drive.

CAUTION

When using wildcard characters within the main part of a file name, be sure to use a period after the wildcard character. Failure to use the period (for example, PROG* without a period) will cause all files to be deleted.

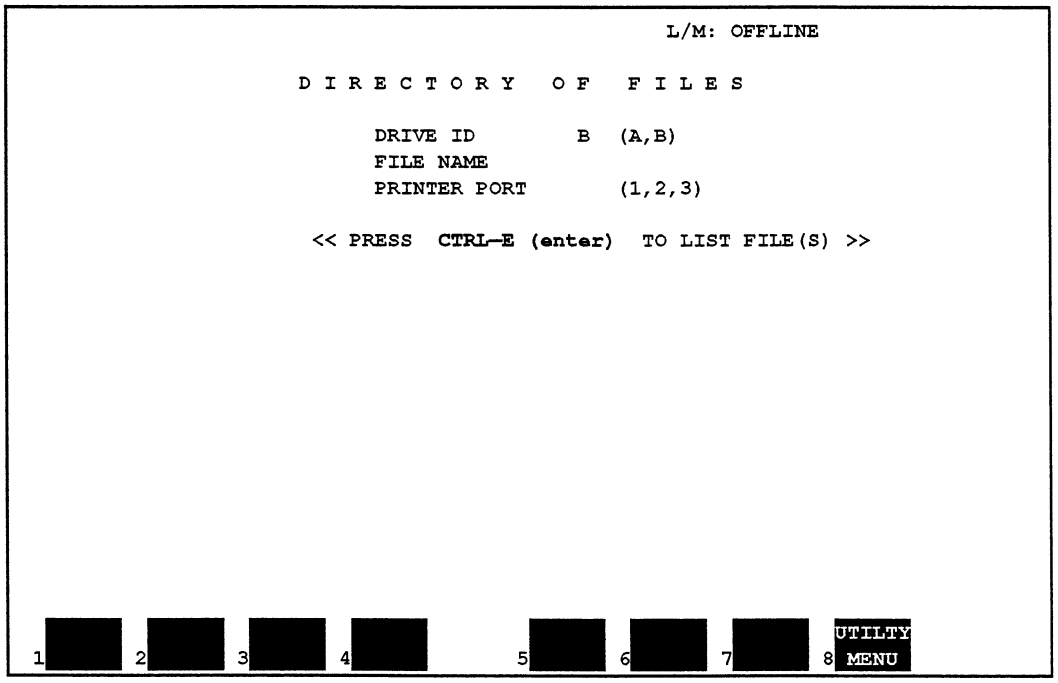
To delete the primary program files (.LAD, .RDF, .NAM, and .EXP files), enter just the file name without an extension.

3. Press CTRL-E (or the Enter key). If one file name with extension (for example, PROGRAM1.TXT) was entered, the file is deleted immediately.
4. If wildcards were used to specify multiple files for deletion, the system asks you if you wish to check each file name before deleting it. Press the Return key if you want to check each file name before the system deletes the file. To have the system automatically delete all files without checking them first, enter **N**.
5. After the last file is found and deleted, the screen displays the message "Deletion completed."

Displaying and Printing a Directory of Files

The Directory utility is used to display a list of the files on a disk. The file listing can also be printed out if a printer has been set up and is on-line to the system. The directory lists all files on the disk. For each file, the listing shows its size in bytes, and the time and date it was last stored on the disk.

To display the Directory of Files, press F5 (Directory of Files) from the Utilities menu.



1. Identify the drive from which to read the files. If the directory source is a diskette drive, insert the diskette into the drive.
2. For a listing of all the files on the source drive, make no entry for file name.
3. For a listing of selected files on the source drive, enter a file name. Use wildcard characters as needed in the file name. For example, enter *.LAD for a listing of all ladder diagram files on the source drive, or (name).* for a listing of all files with the same program name.
4. To print a copy of the listing, enter the number of the printer port.
5. Press CTRL-E (or the Enter key). A listing of files appears on the directory page. The listing has the following format:

```
   FILE              NUMBER OF              LAST MODIFIED
   NAME               BYTES                DATE              TIME

PROGRAM1.LAD           3078               02-18-86          13:19:29
PROGRAM1.LBU           3016               02-06-86          10:46:08

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .


FREE SPACE REMAINING:  336896
```

The directory listing shows the name of each file and its size in bytes. The time and date shown are those recorded by the system when the file was last updated.

The bottom of the screen shows the amount of free space remaining on the source drive.

If the directory includes more files than can be displayed on the screen, the list scrolls upward. To temporarily stop the scroll (and printout), press the Pause/Resume (F2) function key. To resume the scroll (and printout), press the (F2) key again.

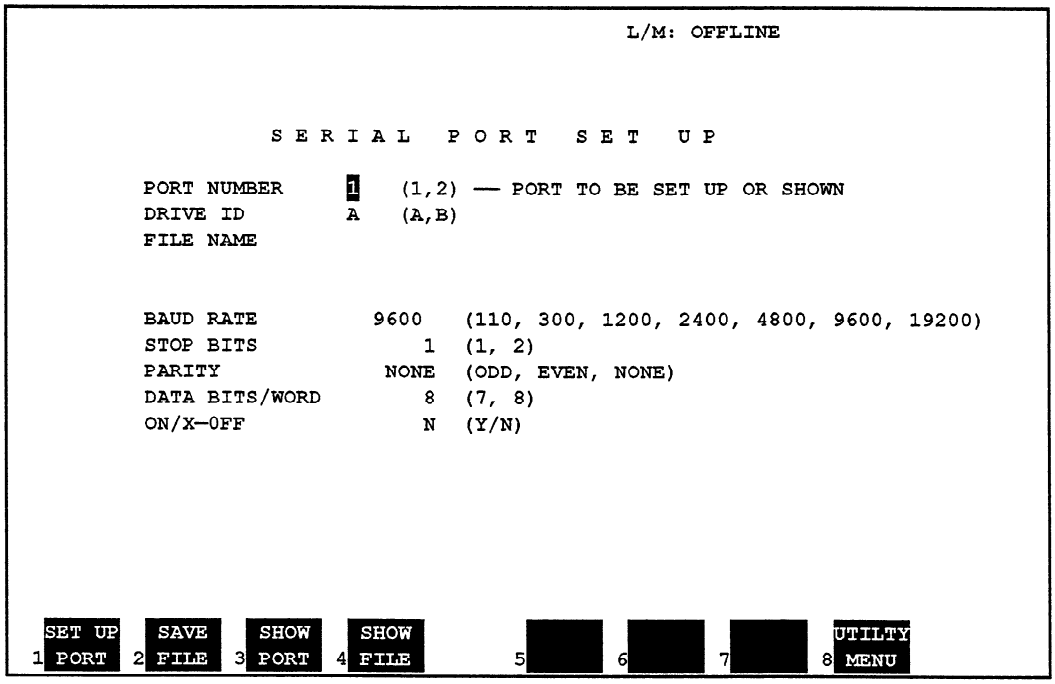
To end a directory listing before it is finished, press the Abort (F4) function key. Press CTRL-E (or the Enter key) to start the directory from the beginning.

To return to the Utilities menu, press the Utility (F8) function key.

Setting Up Serial Ports

The serial ports in the system (designated 1 and 2) can be used for serial printers, tape readers, or CCM RS-232/422 communications. The characteristics of the serial ports must be established before the system can use them to communicate data. Use the Port Set Up utility to assign or display the characteristics of the serial ports in the system.

To display the Serial Port Setup screen, press F6 (Port Setup) from the Utilities menu.



Function Key	Function	Description
F1	Set Up Port	Implement the parameters.
F2	Save File	Create a Port Setup file.
F3	Show Port	Display present port parameters. To determine the current settings for one of the serial ports, enter the number of the port (1 or 2) for PORT NUMBER and press F3.
F4	Show File	Display port parameters from a file. Port Setup values may be stored in a program file with the file name extension .PSU. To determine the current values stored in a Port Setup file, enter the drive where the file is stored for DRIVE ID. Then, enter the name of the file for FILE NAME, and press F4. The current setup characteristics of the file are displayed.

Setting up Port Parameters

Serial port characteristics can be set up for current use only, or they can be stored in a file and loaded into the system each time it is powered up. Follow the steps below to set up the software to communicate over a serial port. If the port is being used for communication with the CCM card in the CPU, the parameters set up must match those of the CCM card. It is also necessary to configure the hardware, as described in appendix A, *Setup Information.*

1. Enter the following information about the device that will be using the serial port:

Baud Rate	The communications rate, in bits per second.
Stop Bits	All communications use one start bit. Slower devices may use two stop bits.
Parity	An ASCII character may consist of either seven or eight data bits. Specify whether parity is indicated by an odd or even number of bits, or whether no parity bit is added to the word.
Data Bits	Specify whether the device recognizes 7 or 8 bit words.
X-ON/X-OFF	Change this entry to **Y** to select Level 2 protocol. Level 2 protocol allows the device connected to the serial port to suspend or restart transmitting characters using the X-ON and X-OFF characters. When the system receives an X-OFF character (DC3-13H) from the device, it stops transmitting characters. When the system receives an X-ON (DC1-11H) character, transmission resumes. The X-OFF character must be sent by the device before the reception of the last data bit of the current character being sent to the system, or the next character will also be sent. If X-ON/X-OFF protocol is used with a device that does not support hardware handshaking, the hardware lines DTR-DSR and RTS-CTS must be tied together (pins 4-5 and 6-9 on the 9-pin port).

2. Enter the number of the serial port being set up.
3. If this setup will be stored in a file for use in future start up, go immediately to step 4.

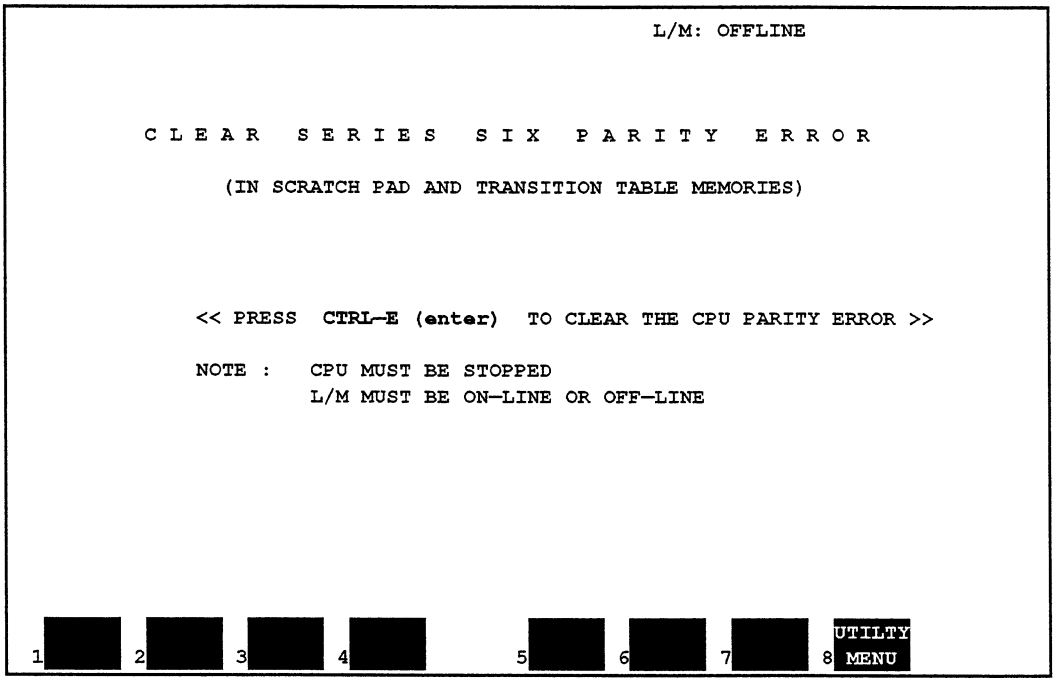
   If this setup is intended only until power is removed from the system, or until new setup information is entered, press F1 (Set Up Port) to execute the setup.
4. If this setup will be stored in a file, place the system diskette in a drive, and enter the letter designation of the drive. The set up file will be written to the diskette, so remove any write-protection from the diskette temporarily. If storing to a hard disk, the file will be stored in the LM6 subdirectory.
5. Enter the file name PORT1.PSU or PORT2.PSU, depending on whether you are setting up port 1 or 2. The file may be stored as any file name.PSU, but must be renamed to either PORT1.PSU or PORT 2.PSU, depending on whether you want to set up port 1 or port 2 automatically at powerup.
6. Press F2 (Save File). The setup file will be placed on the system diskette. However, the setup information in the file will not be used until the file is loaded into RAM during the next powerup.

Clearing CPU Parity Errors

Parity is a type of integrity check performed on memory areas. Most memory areas are reloaded and their parity checked during normal operations, when using the Load/Store/Verify functions. However, this does not clear parity errors in the Scratch Pad or Transition tables.

Parity errors in Scratch Pad Memory and Transition tables are cleared with the Clear CPU Parity Error utility. No data is lost while this utility is in use. The data currently stored in the Scratch Pad and Transition table memories is temporarily stored into system RAM, and loaded back to the CPU at the completion of the utility.

To display the Clear Parity Errors screen, press F7 (Clear Parity) from the Utilities menu.



The CPU must be stopped. The system must not be in Monitor mode.

Press CTRL-E (or the Enter key) to reset the Scratch Pad and Transition table memories. When the utility is completed, the screen displays the message ''SERIES SIX PLC PARITY ERROR CLEARED.''

Clearing All CPU Parity Errors

When starting a CPU for the first time, or after changing a memory module, CPU parity errors must be cleared. If this is not done, when the CPU is started parity errors may be in any of the registers, I/O tables, override tables, scratch pad, and/or transition tables. For example, if an extended register having a parity error is accessed by the Logicmaster 6 system (either through the Display Reference Tables function, or while observing a Move Table Extended in On-Line mode) or by the ladder logic, the CPU will stop. (It is also possible for the CPU to start up without a parity error if Logicmaster 6 software is not connected.)

To clear CPU parity errors, including the scratch pad and transition table memories:

1. Place the system in On-Line mode. Verify that the system contains the correct function level of the CPU (basic, advanced, extended, expanded). From the Supervisor menu, select Scratch Pad (F4). Press F5 (Make Equal) to make these entries agree. Then, return to the Supervisor menu.
2. Clear parity errors in the Scratch Pad and transition table memories using the Clear Parity function. From the Supervisor menu, select Utilities (F8). From the Utilities menu, select Clear Parity (F7). Then press CTRL-E (or the Enter key), and return to the Supervisor menu.
3. Place the system in Off-Line mode. Clear parity errors in table memory by storing the contents of Logicmaster 6 system memory to the CPU. This memory may contain a program to be stored to the CPU, or may first be cleared so that no program is stored to the CPU. From the Supervisor menu, select Load/Store/Verify (F6). To store a program to the CPU, go to step 3A. To clear the CPU table memory, go to step 3B.
   A. If the program to be stored to the CPU is not already in Logicmaster 6 system memory, load it by pressing F1. Specify the drive ID for the program, and enter its name. Then, press CTRL-E (or the Enter key). After the program is loaded into the Logicmaster 6 system, go to step 3C.
   B. If Logicmaster 6 memory currently contains a program that you do not want to store to the CPU, the program must be cleared from memory. If you want to save the program, use the Store function (F2) before clearing Logicmaster 6 memory. Clear Logicmaster 6 memory by pressing F5 (Clear) from the Load/Store/Verify menu. Then, press CTRL-E (or the Enter key), and return to the Supervisor menu. Go to step 3C.
   C. From the Load/Store/Verify menu, press F2 (Store). On the Store Program/Tables screen, enter **P** for CPU as the Drive ID. Then, press CTRL-E (or the Enter key).
4. Start the CPU. To start the CPU using the CPU keyswitch, go to step 4A. To start the CPU from the Logicmaster 6 system, go to step 4B.
   A. To start the function from the CPU, turn the keyswitch on the CPU to Stop, then to Run. Go to step 4C.
   B. To start the function from the Logicmaster 6 system, place the system in On-Line mode. Go to the Supervisor menu, and select Scratch Pad (F4). Then, press F3 (CPU Start). Go to step 4c.
   C. After the function is complete, return to the Supervisor menu.

Chapter 12 Programming

GEK-25379

Chapter 12 presents general programming concepts. Chapters 13 and 14 describe the specific programming functions for the Series Six PLC.

Chapter 12 consists of the following sections:

Section 1. Ladder Logic Programs: describes the CPU scan and discusses the format of a program.

Section 2. The Program Function Sets: summarizes the function sets available in the Series Six PLC.

Section 3. CPU Memory Mapping for Conventional I/O Systems: Refer to Section 3 if your CPU uses Normal I/O addressing. Section 3 explains the I/O and Auxiliary I/O tables, and explains how auxiliary references are mapped into register memory.

Section 4. CPU Memory Mapping for the Series Six Plus PLC: Refer to section 4 if you are using an Expanded Function Set with the Series Six Plus PLC. Section 4 explains how to select Expanded or Normal mode. Charts show how register memory is mapped with the I/O and other functions. Section 4 also describes the Genius I/O fault table and use of the computer mailbox.

Section 5. Reserved References: describes the programming and I/O and register references required for optional modules. When creating a ladder logic program, you must be careful not to use either I/O or register references that are already assigned for use by system modules. In particular, any system which contains one or more of the following must be very carefully designed:

- Smart peripheral cards, such as the ASCII/BASIC Module, Communications Control Module, or LAN Interface Module.
- Data Processor Unit.
- Redundant Processor Unit.
- Advanced I/O Receivers.
- Expanded I/O.
- I/O Transmitter Modules.
- Genius I/O (with diagnostics enabled).
- Series 90-70 I/O Racks.

Refer to section 5 for more information.

SECTION 1
Ladder Logic Programs

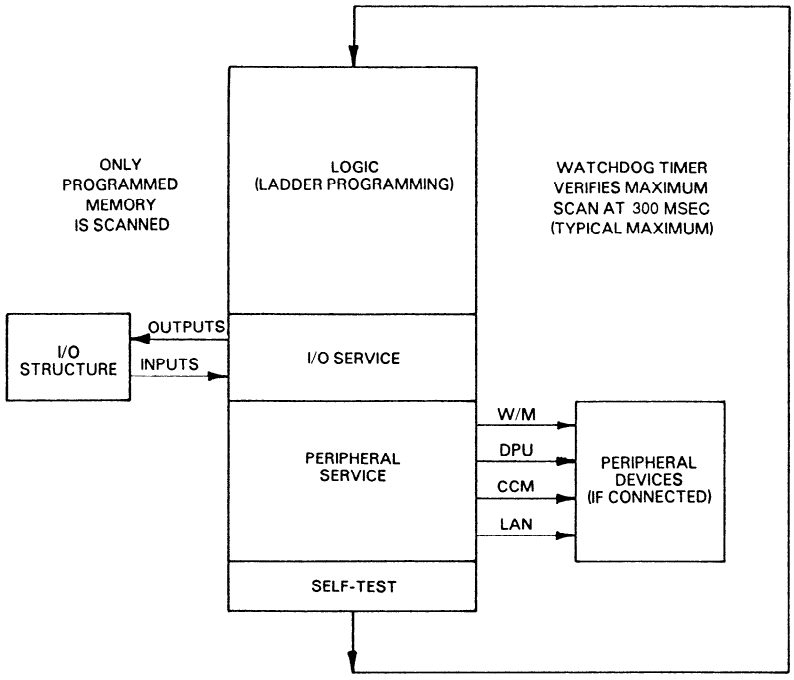
This section is an introduction to ladder logic programs. It explains:

- How the CPU executes a program.
- The format of the ladder logic.
- The elements of a ladder diagram.
- How a program can be edited.
- The general format of a program function.
- Types of program references.

How the CPU Executes a Program

A ladder logic program is a continuous sequence of logic and instructions. The Series Six CPU executes the ladder program as part of its regular scanning cycle.

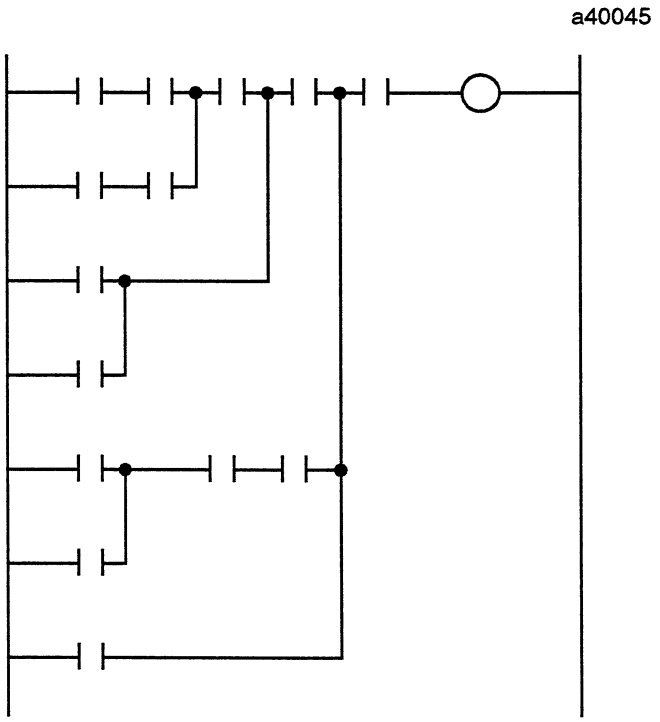
pcs6840265



The CPU begins executing the logic at the top of the ladder and ends at the bottom. After each program scan, the CPU services the I/O. The CPU transfers output status to output modules, and reads the status of input modules. Next the CPU services other devices, such as the Logicmaster 6 system, a GEnet LAN Interface Module, or Communications Control Module (CCM). At the end of the scan, the CPU executes a self-check hardware test. Then, it begins the next scan. The time required for one scan depends on the configuration of the entire system, and the size and complexity of the program. A timer, called a watchdog timer, shuts down the CPU and the I/O if the scan time is too long.

Ladder Diagram Format

The program logic that is executed by the CPU during its regular scan can be represented graphically as a ladder diagram.

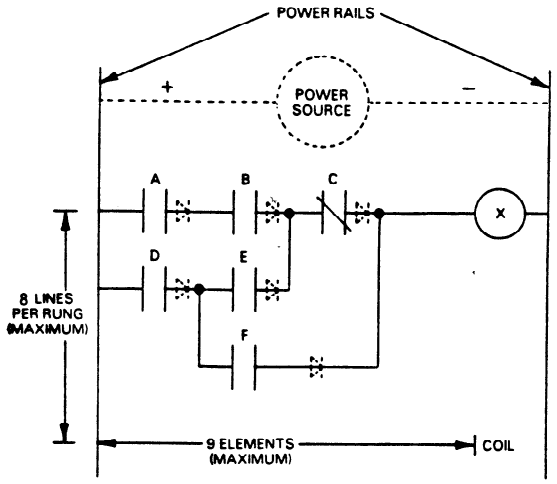


A ladder program consists of a sequence of linked "rungs". Each rung must begin at the left side, which is called a rail. Each rung can include up to 8 *parallel* lines of logic. However, each logic rung may have only one connection to the right rail.

Elements of a Ladder Diagram

Each line of a rung may consist of up to nine program elements, or instructions, entered in *series*, and a coil. Vertical connections can be made between two parallel lines of logic in positions (columns) 1 through 8. Most rungs end in a coil in the tenth column. The coil represents the output of the rung.

A ladder diagram has a symbolic "power source". Power flows from the left rail (+) to the coil connected to the right rail (-). The phantom diodes are implied but not shown. They illustrate that power can flow only from left to right, or up or down.



Using the Edit Program Function to Create Programs

Programs are created using the Edit Program function, which is described in chapter 5, *Edit Program.* The Edit function has a number of special features that make even complex ladder diagrams easy to create. In addition, the Edit Program function is fully supported by Help screens, which you can refer to during editing. The Edit function allows great flexibility in entering program elements. However, it will not allow you to program a rung with incorrect format or syntax.

Illegal Rung: Short circuit

```
+--] [--+--] [--+--] [--+--] [--+-------+-------+-------+-------+-------+-( )-+
                |       |
                +-------+
```

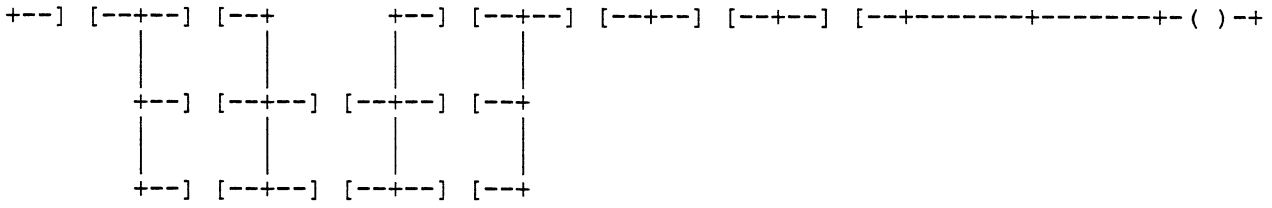
Illegal Rung: Nothing allowed in parallel with a mnemonic

```
+--] [--+--] [--+--] [--+--] [--+--] [--+[xxxxx]+-------+-------+-------+-( )-+
                                       |       |
                                       +--] [--+
```

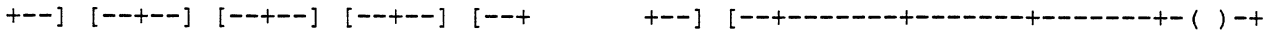
Illegal Rung: No more than one output per rung

```
+--] [--+--] [--+--] [--+--] [--+--] [--+--] [--+--] [--+-------+-------+-( )-+
                                                |
                                                +-------+-------+-------+-( )-+
```

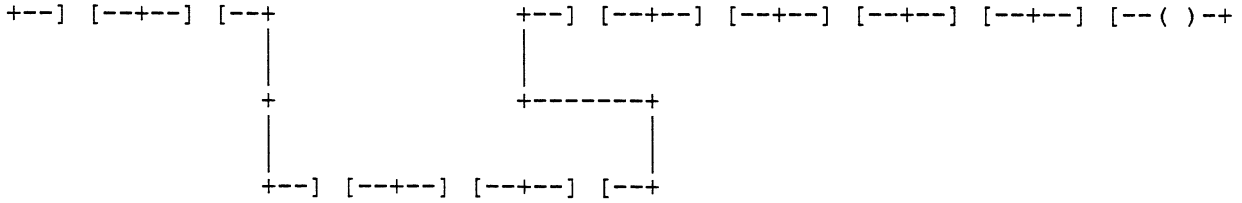
Illegal Rung: Connect contact to topmost available junction



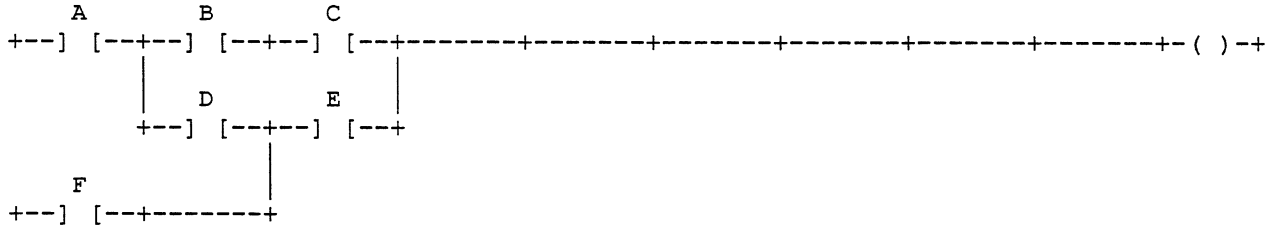
Illegal Rung: Open circuit



Illegal Rung: More than nine series contacts



Illegal Rung: Starting a branch within a branch



Format of a Ladder Diagram Function

Whether simple or complex, the functions that make up a line of logic have a similar format.

```
 I00145                 _______ Reference (an input)
---| |---               _______ Function (a contact)

 O00001                 _______ Reference (an output)
---( )                  _______ Function (relay coil)

  R0004                 _______ Reference (a register)
-|PRERG|-               _______ Function (the mnemonic for Preset Register)

  Const                 _______ Constant
-|PRESC|-               _______ Function (the mnemonic for Preset Register)
    012                 _______ The value of the constant

  R0001    R0002    R0003       ____ References (three registers)
-|  A    +   B    =    C   |-    ____ Function (the mnemonic for Binary Add)
```

Each function is represented by a symbol or "mnemonic". For example:

```
---|/|---                symbol for a normally-closed contact

-| BIN  TO  BCD  |-      mnemonic for Binary to BCD Conversion function
```

User References

The values used by the functions in a program are either constants or register or discrete references. During programming, the place for the reference appears on the screen directly above the symbol or mnemonic for the function. A constant is a set value that is entered during programming and not changed by program execution. An example of a constant is shown above, with the mnemonic PRESC.

A discrete or register reference is a memory location that contains a status or value that may change during program execution. The mnemonic PRERG above is an example. A reference may be an I/O or Auxiliary I/O address, or a register. Some functions use a sequence of adjacent memory locations to store data. Only the first reference is entered into the ladder diagram. This is called an "explicit reference". The other addresses that are used by the function, but do not appear in the ladder diagram, are called "implicit references".

The range of valid references that can be used depends on the function level of the CPU. Before programming, the Scratch Pad screen should be set to this function level. After this is done, the Logicmaster 6 software will not allow entry of program references that are out of range for the CPU.

A few program functions have limited reference ranges, regardless of the CPU function level. For example, one shots and counters can only be used in the Main I/O status table. The software will not permit incorrect references to be entered for any function.

Program Functions

The functions that can be used in a ladder logic program depend on the function level of the CPU that will execute the program. The function level may be:

Function Level	CPU Microcode Version
Basic/1.0	8
Basic/1.01	9
Extended/2.0	101
Extended/2.01	102
Extended/2.02	103
Advanced/3.0	104
Advanced/3.01	105
Advanced/3.02	106
Expanded/4.0	110
Expanded/4.10	120
Expanded II/5.0	130
Expanded II/5.03	133

If you are not sure what the function level is, use the Scratch Pad display, as described in chapter 3, *Scratch Pad.* Each function set includes all the features of the earlier function sets, plus the additional features summarized below.

The Basic function set provides all the basic elements for a ladder diagram, including contacts, coils, unsigned binary arithmetic, and other program functions.

The Extended function set provides additional capabilities, including Auxiliary I/O references, signed double precision arithmetic, subroutines, table functions, and matrix functions.

The Advanced Functions, which are an enhancement of the Extended function set, add the DPREQ for ladder windowing and the Status command for I/O parity check modification.

The Expanded function set adds 16K of I/O points, 16K of directly-addressable registers, floating point arithmetic, Expanded functions (Do I/O, Status, and Windowing), and Genius diagnostics.

The Expanded II function set provides additional capabilities, including 64K user program memory, overrides of Auxiliary I/O, special programming interface to Series 90-70 I/O racks, and a user program checksum displayed on the Scratch Pad screen.

If the Logicmaster 6 software is powered up in Off-Line operating mode, it defaults to use of the Expanded II function set. If the CPU does not have Expanded II functions, this entry should be changed on the Scratch Pad screen before beginning the program.

SECTION 2
CPU Memory Mapping for Conventional I/O Systems

This section describes how the Series Six CPU stores discrete and register references in memory. This is called "memory mapping". You will need this information to assign program references.

Many of the functions in a program can reference the Input and Output status tables and register memory in the CPU. Certain hardware (CPU) tables are also used by the functions. These tables and the various memories are summarized below. All tables are retained during power failure.

Input Table

There are 1024 inputs available for reference from 1 to 1024. The Input table consists of a group of consecutive memory locations. Inputs 1001 to 1024 cannot be connected to real input devices. However, they can be used for internal storage and can be used for program references.

In the program, inputs are used to represent contacts activated by devices such as pushbuttons, limit switches, analog sensors or relay contacts. Inputs that represent actual hardware devices are called "real" inputs.

Inputs also represent contacts operated by output coils of program functions, such as relays, counters, timers, latches, and one-shots. Input references that do not represent actual hardware inputs are called "internal" inputs.

Output Table

There are 1024 outputs available for reference from 1 to 1024. The Output table consists of a group of consecutive memory locations. Outputs 1001 to 1024 cannot be connected to real devices. However, they can be used for internal storage and can be used for program references.

In the program, outputs can be used to represent the coils of relays, counters, timers, latches, and one-shots. Like inputs, there are also real and internal outputs.

Transition Table

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

The other half of the Transition table is used by the auxiliary counter and one-shot functions with the advanced functions. These 1024 bits correspond to the lowest 64 words (x16 bits) of the register memory (which is the Auxiliary Output table).

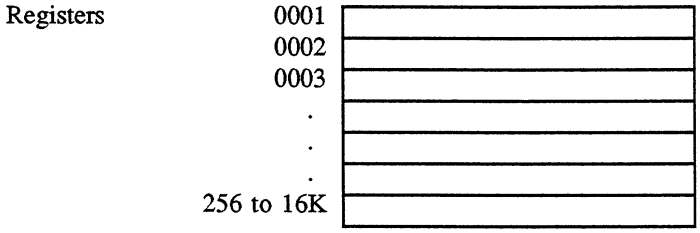
The only condition under which the Transition table can be written into or read by the Logicmaster 6 system is when the table is cleared of parity errors.

Override Tables

The Override tables store override information for I/O references in the Main and Auxiliary I/O chains. The Override tables are used by the following functions: relays, counters, timers, latches, and one-shots. Mnemonic functions ignore the Override tables. When an input, output, or auxiliary I/O reference is overridden, it is maintained in the state (on or off) it was in when the override was applied. This state remains the same until the override is removed or its state is changed. Overridden bits can be changed by the mnemonic functions. References can be overridden, as described in chapter 7, *Display Reference Tables*. Expanded Inputs and Outputs cannot be overridden.

Register Memory

The term "register" refers to a group of 16 consecutive bits located in register memory. The structure of these registers is fixed. Each register is numbered, beginning at 0001. The number of registers that are available depends on the amount of register memory available in the CPU.

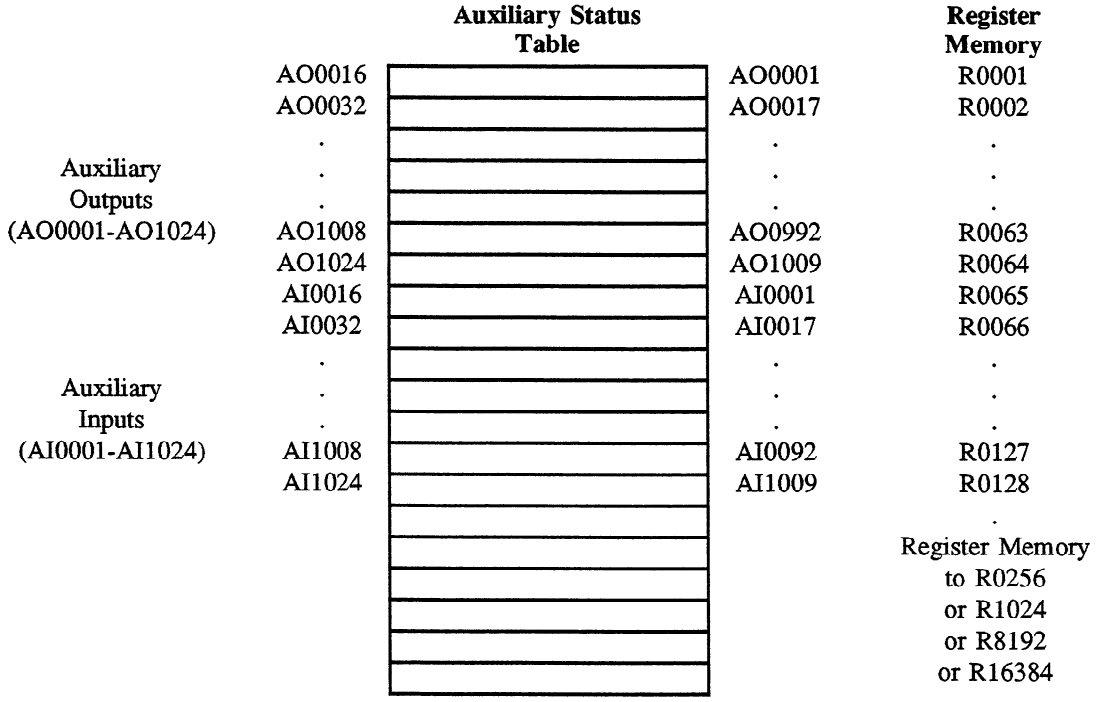


Register memory consists of from 256 to 16K consecutive 16-bit storage locations, depending on the CPU. Some of the mnemonic functions refer to locations in the register memory and use those locations for data manipulation when specifying lists, storing results of arithmetic operations, data tables, data moves, and matrixes.

Auxiliary References

Auxiliary references provide an additional 1000 I/O references. Auxiliary I/O is mapped to register memory. The Auxiliary I/O can be referenced as bits within their assigned registers, or by their AI or AO references. This dual reference capability improves efficiency, and can be a valuable programming tool.

The next illustration shows the assignment of Auxiliary I/O to register memory.



Auxiliary I/O are mapped to registers R0001 through R0128. This makes it very convenient to translate 16 discrete references into a single register, or to refer to register bits as discrete references in a program.

Auxiliary outputs are mapped from R0001 to R0064. Auxiliary inputs are mapped from R0065 to R0128.

SECTION 3
CPU Memory Mapping for the Series Six Plus PLC

This section describes how the Series Six Plus PLC maps I/O to memory when Expanded I/O addressing is used. You will need this information to assign program references.

Either Normal or Expanded I/O addressing can be used for a Series Six Plus PLC. The addressing mode selected depends on the hardware configuration of the system. If Expanded addressing is selected, a bus controller or I/O Transmitter Card is needed on every channel if more than one channel is enabled.

The I/O addressing for a system must be set up in hardware, as described later in this chapter. The ladder logic program must reflect the hardware configuration. For the program, selection of Normal or Expanded addressing is made on the CPU Configuration menu. This is described in chapter 10, *Expanded Functions Menu.*

Normal I/O Addressing

If Normal I/O addressing is used, the Series Six Plus CPU scans the Main I/O chain and the Auxiliary I/O chain in the same manner as other Series Six CPUs.

The maximum number of I/O points available in Normal mode is 4000:

1000 inputs and 1000 outputs in the Main I/O chain.
1000 inputs and 1000 outputs in the Auxiliary I/O chain.

Each chain actually has 1024 inputs and outputs; however, only the first 1000 are physical I/O points. The rest can be used as internal I/O in a program.

The CPU stores the ON and OFF states from the Main I/O chain in the I/O status table.

The CPU stores the ON and OFF states from the Auxiliary I/O chain in the Auxiliary I/O status table.

The Auxiliary status table is mapped into the first 128 words of register memory. R0001 to R0064 contain the Auxiliary Output table, and R0065 to R0128 contain the Auxiliary Input table.

Expanded I/O Addressing

If Expanded I/O addressing is used, the Series Six Plus PLC maps Expanded I/O points to register addresses. The range of I/O available depends on CPU register memory size. The charts on the following pages show actual memory usage for 256, 1K, 8K, and 16K register memory.

In Expanded I/O addressing, real I/O points for channel 0 are scanned on the Main I/O chain, and their status is maintained in the Main I/O status table.

Main I/O and Auxiliary I/O are mapped into the I/O and Auxiliary AI/AO tables, respectively, as always. They are hardware-configured to be channel 0 and channel 8, respectively, on the I/O chain. No program reference is made to channel 0+ or channel 8+.

Other channels of I/O are mapped into registers as shown in the charts in this section.

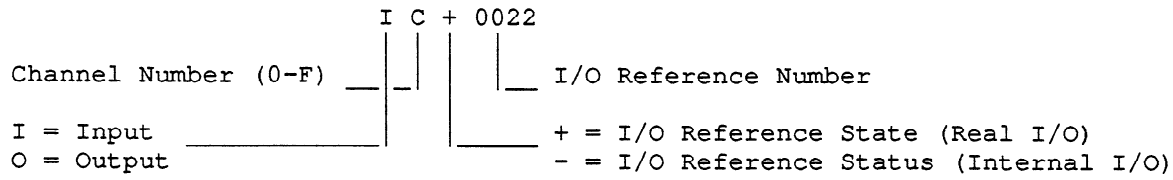
I/O points in the Expanded mode are selected in 1K increments, referred to as channels. Up to 14 channels (7 on the Main I/O chain, 7 on the Auxiliary I/O chain) can be configured within a range of stop and start addresses specified using the CPU Configuration function. The Main I/O chain channels are 1 to 7, and the Auxiliary chain channels are 9 through F. The total real I/O points available through

the use of both chains plus the Main I/O and Auxiliary I/O tables is, therefore, 16K inputs and 16K outputs.

The format for addressing I/O in the Expanded mode must include:

- A channel number (1-7, 9-F).
- Either I or O, and a real I/O or internal I/O reference identifier.
- The I/O number.

This format is shown by the example below.



Thus, the format for real world I/O points for channel 3 on the Main I/O chain is I3+0001 to I3+1024 for inputs and O3+0001 and O3+1024 for outputs.

When referencing an I/O point in a program, a prefix must properly address the applicable I/O chain. For example: I3+0101, O3+0101, IB+0234, or OB+0567.

Each 1K channel requires 128 words of memory, 64 for inputs and 64 for outputs (64 words x 16 bits = 1024 I/O). Although 1024 bits are available in each channel, only 0001 to 1000 are used for actual physical I/O points. 1001 to 1024 are reserved.

Expanded I/O Addressing for 8K and 16K Registers

The table below shows how Expanded I/O is mapped into memory for a CPU with either 8K or 16K of register memory.

- Do not use Real I/O references O0+0001 to O0+1024, I0+0001 to I0+1024, O8+0001 to O8+1024, and I8+0001 to I8+1024 as program references. (Release 4.0 and later of Logicmaster 6 software will not allow these references to be used.)
- Internal I/O references O0-0001 to O0-1024, I0-0001 to I0-1024, O8-0001 to O8-1024, and I8-0001 to I8-1024 can be used.
- Channel 1-7 real I/O references are for the Expanded Main I/O channels.
- Channel 9-F real I/O references are for the Expanded Auxiliary I/O channels.

Register Range	I/O Addresses	Register Contents	
	Real I/O		
R00001 - R00064	AO0001 - AO1024		Auxiliary I/O
R00065 - R00128	AI0001 - AI1024		
R00129 - R00192	O1+0001 - O1+1024	Main Chain:	Channel 1+
R00193 - R00256	I1+0001 - I1+1024		
R00257 - R00320	O2+0001 - O2+1024		Channel 2+
R00321 - R00384	I2+0001 - I2+1024		
R00385 - R00448	O3+0001 - O3+1024		Channel 3+
R00449 - R00512	I3+0001 - I3+1024		
R00513 - R00576	O4+0001 - O4+1024		Channel 4+
R00577 - R00640	I4+0001 - I4+1024		
R00641 - R00704	O5+0001 - O5+1024		Channel 5+
R00705 - R00768	I5+0001 - I5+1024		
R00769 - R00832	O6+0001 - O6+1024		Channel 6+
R00833 - R00896	I6+0001 - I6+1024		
R00897 - R00960	O7+0001 - O7+1024		Channel 7+
R00961 - R01024	I7+0001 - I7+1024		
R01025 - R01088		(user registers)	
R01089 - R01152			
R01153 - R01216	O9+0001 - O9+1024	Auxiliary I/O:	Channel 9+
R01217 - R01280	I9+0001 - I9+1024		
R01281 - R01344	OA+0001 - OA+1024		Channel A+
R01345 - R01408	IA+0001 - IA+1024		
R01409 - R01472	OB+0001 - OB+1024		Channel B+
R01473 - R01536	IB+0001 - IB+1024		
R01537 - R01600	OC+0001 - OC+1024		Channel C+
R01601 - R01664	IC+0001 - IC+1024		
R01665 - R01728	OD+0001 - OD+1024		Channel D+
R01729 - R01792	ID+0001 - ID+1024		
R01793 - R01856	OE+0001 - OE+1024		Channel E+
R01857 - R01920	IE+0001 - IE+1024		
R01921 - R01984	OF+0001 - OF+1024		Channel F+
R01985 - R02048	IF+0001 - IF+1024		

Register Range	I/O Addresses	Register Contents	
	Internal I/O		
R02049 - R02112	O0-0001 - O0-1024	Main Chain:	Channel 0-
R02113 - R02176	I0-0001 - I0-1024		
R02177 - R02240	O1-0001 - O1-1024		Channel 1-
R02241 - R02304	I1-0001 - I1-1024		
R02305 - R02368	O2-0001 - O2-1024		Channel 2-
R02369 - R02432	I2-0001 - I2-1024		
R02433 - R02496	O3-0001 - O3-1024		Channel 3-
R02497 - R02560	I3-0001 - I3-1024		
R02561 - R02624	O4-0001 - O4-1024		Channel 4-
R02625 - R02688	I4-0001 - I4-1024		
R02689 - R02752	O5-0001 - O5-1024		Channel 5-
R02753 - R02816	I5-0001 - I5-1024		
R02817 - R02880	O6-0001 - O6-1024		Channel 6-
R02881 - R02944	I6-0001 - I6-1024		
R02945 - R03008	O7-0001 - O7-1024		Channel 7-
R03009 - R03072	I7-0001 - I7-1024		
R03073 - R03136	O8-0001 - O8-1024	General use or	Channel 8-
R03137 - R03200	I8-0001 - I8-1024		
R03201 - R03264	O9-0001 - O9-1024	Auxiliary I/O:	Channel 9-
R03265 - R03328	I9-0001 - I9-1024		
R03329 - R03392	OA-0001 - OA-1024		Channel A-
R03393 - R03456	IA-0001 - IA-1024		
R03457 - R03520	OB-0001 - OB-1024		Channel B-
R03521 - R03584	IB-0001 - IB-1024		
R03585 - R03548	OC-0001 - OC-1024		Channel C-
R03549 - R03712	IC-0001 - IC-1024		
R03713 - R03776	OD-0001 - OD-1024		Channel D-
R03777 - R03840	ID-0001 - ID-1024		
R03841 - R03904	OE-0001 - OE-1024		Channel E-
R03905 - R03968	IE-0001 - IE-1024		
R03969 - R04032	OF-0001 - OF-1024		Channel F-
R04033 - R04096	IF-0001 - IF-1024		
	Other References		
R04097 - R04112		Bus Controller Bit Map	
R04113 - R04116		User Registers	
R04117 - R04119		Real Time Clock	
R04120		Fault Table Pointer	
R04121 - Rxxxxx		Fault Table Entries	
R08123 - R08192 (8K)		Computer Mailbox for 8K	
R16315 - R16384 (16K)		Computer Mailbox for 16K	

Expanded I/O Addressing for 1K Registers

The table below shows how Expanded I/O is mapped into memory for a CPU with 1K register memory.

- Channel 0 I/O is scanned on the Main I/O chain.
- Expanded discrete references O1+0001 to O7+1024 and I1+0001 to I7+1024 are mapped into the register table.
- Auxiliary discrete references AO0001 to AO1024 and AI0001 to AI1024 are mapped into registers.

Register Range	I/O Addresses	Register Contents
R0001 - R0128	(AO0001 - AO1024) (AI0001 - AI1024)	Auxiliary I/O
R0129 - R0256	(O1+0001 - O1+1024) (I1+0001 - I1+1024)	Genius I/O point fault status If enabled, associated with Main Chain I/O, 0001 - 1024
R0257		Bus Controller Status Bit Map
R0258 - R0266	User Registers	If Genius I/O diagnostics is enabled, can be referenced as:
R0267 - R0269	Real Time Clock	
R0270	Fault Table Pointer	O2+0001 - O2+1024
		I2+0001 - I2+1024
R0271 - Rxxxx	Fault Table Entries	thru
		O7+0001 - O7+1024
R0955 - R1024	Computer Mailbox	I7+0001 - I7+1024

Expanded I/O Addressing for 256 Registers

The table below shows how Expanded I/O is mapped into memory for a CPU with 256-word register memory.

- Auxiliary and Expanded discrete references AO0001 through AO1024, A10001 through AI1024, O1+0001 through O1+1024, and I1+0001 through I1+1024 are mapped into the register table.

Register Range	I/O Addresses	Register Contents
R0001 - R0128	(AO0001 - AO1024) (AI0001 - AI1024)	Either Auxiliary I/O or general use registers.
R0129 - R0160	(O1+0001 - O1+0512)	If Genius I/O diagnostics is enabled, can be referenced as:
R0161 -	Bus Controller Status Bit Map	O1+0513
R0162 - R0166	User Registers	
R0167 - R0169	Real Time Clock	thru
R0170 -	Fault Table Pointer	
R0171 - Rxxxx	Fault Table Entries	
Rxxxx - R0186	User Registers	O1+1024
R0187 - R0256	Computer Mailbox	(R0193) I1+0001 - I1+1024

Expanded Real-Time Clock

The Series Six Plus CPU maintains a three-word real-time clock in three consecutive registers. The real-time clock is available (enabled) whenever a CONFIG opcode is present in the ladder program.

The exact register address is determined by the total register size of the CPU. The clock is located in the following registers:

Number of Registers	Location of the Clock
8K and 16K	R4117, R4118, R4119
1K	R0267, R0268, R0269
256	R0167, R0168, R0169

The real-time clock maintains an accurate record of the time when Genius I/O faults occur.

The clock tracks hundreds-of-days/days/hours/minutes/seconds/tenths-of-seconds. Each fault is time-stamped with the clock reference, and stored in the Genius I/O fault table if it is enabled. The clock stops when power is removed and must be reset by the user when power is restored. The clock is updated once per sweep and has a cumulative accuracy of 8 seconds per day.

The format for storing the clock data in the three registers is shown below. Data entered in each byte of each register is two BCD digits.

Register	Most Significant Byte	Least Significant Byte
Rxxxx	Seconds (00-59)	Tenths of Seconds (00-09)
Rxxxx +1	Hours (00-23)	Minutes (00-59)
Rxxxx +2	Hundreds of Days (00-99)	Days (00-99)

Genius I/O Fault Table

When Genius I/O diagostics are enabled in the CPU Configuration menu, the Series Six Plus CPU will use register memory space for a table to store Genius I/O faults. The fault table can be as long as desired, up to the number of registers that are available in the CPU above the fault table pointer. The maximum size of the fault table depends on the number of available registers in memory. Each fault requires 10 registers of memory.

The fault table has a default length of 8 faults. The length of the fault table can be changed on the CPU Configuration Setup menu (part of the Expanded functions). For more information, *Expanded Functions Menu.*

Fault Table Pointer

The first register in the fault table is a pointer. It keeps track of the number of faults currently stored in the table. When the table is full, no additional entries are made until the count in the pointer is cleared or reduced by the ladder logic program. Note that the actual content of the table is not cleared to zero; only the pointer (first) register must be changed by the program. New data is loaded into the first available location, with the oldest data "on top". That is, the oldest fault data is stored in the lowest-numbered register after the pointer location.

For example, a total of 27 faults could be stored in a fault table which was 271 registers in length. That would be 10 registers per fault and 1 register for the pointer.

Using Alarm Master™ Software to Monitor Genius I/O Faults

Alarm Master™ is a software package that can be used to create fault-monitoring programs for Genius I/O. These programs run on an Operator Interface Terminal. The Alarm Master software gets its information from the Genius I/O fault table. It can either copy faults from the table or remove them. If Alarm Master removes faults from the table, the Logicmaster 6 display of Genius I/O faults from the table will not be correct. When a system uses Alarm Master, the interaction between the Alarm Master program and the Logicmaster 6 software should be carefully planned. For more information, refer to the *Alarm Master User's Manual* (GEK-96669).

Fault Table Entries

Each of the fault entries in the fault table is composed of 10 registers. The space in these registers is apportioned, as explained in the following table.

Register 1	Least Significant Byte Most Significant Byte	IOC address IOC address
Register 2	Least Significant Byte Most Significant Byte	I/O address I/O address
Register 3	Least Significant Byte Most Significant Byte	Circuit/channel number IOC status byte 3
Register 4	Least Significant Byte Most Significant Byte	IOC status byte 4 IOC status byte 5
Register 5	Least Significant Byte Most Significant Byte	IOC status byte 6 Fault type
Register 6	Least Significant Byte Most Significant Byte	Fault type Fault description
Register 7	Least Significant Byte Most Significant Byte	Fault description Fault description
Register 8	Least Significant Byte Most Significant Byte	Fault time - tenths of a second Fault time - seconds
Register 9	Least Significant Byte Most Significant Byte	Fault time - minutes Fault time - hours
Register 10	Least Significant Byte Most Significant Byte	Fault time - days Fault time - hundreds of days

Fault Register 1 - IOC or Bus Controller Address

The first register in the fault entry contains the I/O address of the IOC that reported the fault. The address is reported in the following format:

Bit Number	Description
12 - 15	I/O channel containing IOC (range 0 - 15).
10 - 11	Zeros.
0 - 9	I/O address where IOC begins (range 0 - 1023).

This data is derived from the IOC location bit map, as specified by the user in the Configuration op code.

Fault Register 2 - I/O Address

The second register in the fault entry contains the I/O address of the I/O point (or beginning address of the analog channel) that had the fault. If the fault was not related to a specific point or block (i.e., IOC NOT OK or serial bus error), this address will not be applicable.

The address will be stored in the following format:

Bit Number	Description
12 - 15	I/O channel (range 0 - 15).
11	Set to 1 on output point circuit faults, or on block fault of block containing outputs.
10	Set to 1 on input point circuit faults, or on block fault of block containing inputs.
0 - 9	I/O address of failed point (range 0 - 1023).

The data for bits 0 - 9 is derived from IOC status bytes 3 and 4, from the IOC reporting the fault. Bits 8 and 9 are the two least significant bits of IOC status byte 4. Bits 0 - 7 are status byte 3. Bits 10 and 11 are from bits 0 and 1, respectively, of status byte 2. Bits 12 - 15 are the I/O channel containing the IOC reporting the fault.

Fault Register 3 Lower - Circuit or Channel Number

The first byte of the third register of the fault entry is devoted to the circuit or channel number. This byte contains the offset from the beginning of the smart block's I/O addresses to which the I/O address in the previous register corresponds. This allows you to derive the starting address of the smart block containing the fault. This field is only updated if the fault is a circuit fault; otherwise, it is filled with zeros.

Bit Number	Description
0 - 3	Circuit or channel offset (range 0 - 15).
4 - 7	Unused - set to zero.

This data is derived from bits 4 - 7 (the upper four bits) of IOC status byte 5.

Fault Register 3 Upper - IOC or Bus Controller Status Byte 3

The second (upper) byte of fault register 3 is an exact copy of IOC status byte 3. A copy of IOC status bytes 3, 4, 5, and 6, is provided for use in the ladder logic program and/or application program. Any block types or block error types for which the Series Six PLC cannot decode and format a fault table entry may be encoded by interpretting IOC status bytes 3 through 6.

Fault Register 4 - IOC or Bus Controller Status Bytes 4 and 5

IOC status bytes 4 and 5 from the IOC reporting the error are copied into fault register 4. Copies of these IOC status bytes are provided for future Genius faults, and interpretation of these faults by the user's ladder logic program and/or application program.

Fault Register 5 Lower - IOC or Bus Controller Status Byte 6

The first byte of fault register 5 contains IOC status byte 6. When coupled with IOC status bytes 3, 4, and 5, any Genius block fault may be decoded.

Presently, IOC status bytes 3 through 6 will be needed when an IOC reports an I/O point address conflict. In that case, the serial bus address of one of the conflicting modules will be in IOC status byte 5 and the other in byte 6.

Fault Register 5 Upper - Fault Type

The most significant byte of fault register 5 used to store the fault type. This fault data is stored as follows:

Bit Number	Description
0	IOC not okay (no response).
1	Serial bus error.
2	Circuit fault.
3	Loss of module.
4	Addition of module.
5	I/O address conflict.
6	EEPROM failure.
7	Unused, set to 0.

IOC NOT OK: This bit will be set to a 1 when the IOC has failed to respond with input data on the previous I/O scan. The IOC should send a 1 to the PLC in bit 0 of IOC status byte 1 during each I/O scan. The PLC verifies that bit 1 in the input table is a 1; then, resets it each time it performs the diagnostics.

If bit 1 is found to not be a 1, the IOC failed to provide input data. The PLC then decides if this is the first time that it found the IOC not responding. Byte 5 of the IOC control output data is reserved for the use of the PLC for Genius I/O diagnostics. Each time the IOC OK bit is examined, its state is transferred to bit 0 of byte 5 of the IOC control output data. When the IOC OK bit is 0 after the I/O scan, bit 0 of byte 5 of the IOC control output data is also checked. If it is a 1, the IOC NOT OK condition has just occurred for the first time and should be reported in the fault table. If that bit is a 0, the fault has occurred previously and should not be reported again. The PLC should also disregard all other error indications from an IOC while that IOC is not sending IOC OK.

Serial Bus Error: This bit is set to 1 when the IOC status byte 1, bit 1, is a 1 during diagnostics. The serial bus error condition must record its past state in bit 1 of IOC control byte 5, just as the IOC NOT OK error does in bit 0, and must only record a fault on the leading edge of the error condition.

Circuit Fault: This bit is set to 1 when the IOC status byte 1, bit 2, is 1 during the diagnostics.

Loss of a Module: This bit is set to 1 when the IOC status byte 1, bit 3, is 1 during diagnostics.

Addition of a Module: This bit is set to 1 when the IOC status byte 1, bit 4, is 1 during diagnostics.

Point Address Conflict: This bit is set to 1 when the IOC status byte 1, bit 5, is 1 during diagnostics.

EEPROM Failure: This bit is set to 1 when the fifth byte of the IOC status is equal to 00, and the circuit fault bit (bit 2) of IOC status byte 1 is 1, and bit 3 of IOC status byte 6 is 1.

This module fault is singled out from the other types of module faults because it is different in runtime impace and in required corrective action. The EEPROM fault may be diagnosed with no effect on module operation until power is lost. Therefore, this failure is probably a lower priority than other types of module failures. Also, the EEPROM is physically located in the terminal block assembly of the Smart Block. The other module faults are repaired by replacing the electronics module.

Fault Register 6 Lower - Fault Type

The least significant byte of fault register 6 will contain the block type, if applicable, of the block reporting the error. This fault data is stored as follows:

Least Significant Byte Register 6 (Bits 3 - 0)	Fault (Block) Type
0000	Block headend fault.
0001	Discrete circuit.
0010	Analog circuit.
All other patterns	Undefined, but will be copied as found.
Bits 4 - 7	Unused, set to 0.

Fault (Block) Type: If the error reported in the Most Significant Byte of register 5 indicates a circuit fault, then the lower four bits of IOC status byte 5 contain the information as to the type of fault that had occurred. These bits are copied into bits 0 - 3 of the Least Significant Byte of register 6 in the fault table entry. If the error was not a circuit fault, the lower four bits of IOC status byte 5 are still copied into register 6 of the fault table entry, but they do not necessarily indicate fault type information.

Fault Register 6 Upper - Fault Description

Three bytes of the fault table entry are used for the fault description. These are the most significant byte of register 6 and all of register 7. These 24 bits are the module fault types decoded to a single bit for each type of fault.

The individual fault type bits for discrete point blocks are located in the most significant byte of fault register 6. The table below lists the assignments of the fault types to the bits in upper fault register 6.

Bit Number	Description
8	Loss of power.
9	Short circuit.
10	Overload.
11	No load/open line.
12	Over temperature.
13	Switch failed.
14	Undefined (0).
15	Undefined (0).

If IOC status byte number 1 bit 2 is set, indicating a circuit fault, and if byte 5 indicates a discrete circuit fault, then byte 6 of the IOC status should be copied to the most significant byte of register 6 and register 7 should be cleared.

Fault Register 7 - Fault Description

The individual fault type bits for analog blocks are located in the least significant byte of fault register 7. The table below lists the assignments of the fault types to the bits in lower fault register 7.

Bit Number	Description
0	Analog input low alarm.
1	Analog input high alarm.
2	Analog input under range.
3	Analog input over range.
4	Analog input open wire.
5	Analog output under range.
6	Analog output over range.
7	Undefined (0).

The upper byte of fault register 7 is currently undefined and is reserved for future use. As such, it is always filled with zeros.

Bit Number	Description
8	Undefined (0).
9	Undefined (0).
10	Undefined (0).
11	Undefined (0).
12	Undefined (0).
13	Undefined (0).
14	Undefined (0).
15	Undefined (0).

If a circuit fault is indicated, but byte 5 indicates an analog circuit fault, then byte 6 of the IOC status should be copied to the least significant byte of fault register 7 and the most significant bytes of registers 6 and 7 should be cleared. If the fault is of any other type, register 7 and the most significant byte of register 6 should be cleared.

Fault Registers 8, 9, and 10 - Real Time Clock

These three registers contain a real time clock maintained by the CPU's exec program. (See the above register memory maps for the register address vs register memory size.)

The clock will be maintained at all times, whether the CPU is in RUN or STOP mode. If the CPU loses power, the clock loses the amount of time that the CPU is without power. The clock is updated once per sweep.

The format for the storage of the data in the three registers is listed in the following table.

Register	Most Significant Byte	Least Significant Byte
First	Seconds (00 - 59) 2 BCD digits	Tenth seconds (99 - 09) 2 BCD digits
Second	Hours (00 - 23) 2 BCD digits	Minutes (00 - 59) 2 BCD digits
Third	Hundreds of days (00 - 99) 2 BCD digits	Days (0 - 99) 2 BCD digits

Computer Mailbox

The computer mailbox is an automatic communications window that can be directed to a resident Series Six window device. It uses the last 70 consecutive registers out of the total number available. The location of the computer mailbox registers depends on the CPU register memory size. For example, for a CPU with 8K register memory, the computer mailbox is located from R8123 through R8192. For a CPU with 16K register memory, the computer mailbox is located from R16315 through R16384.

Use of the computer mailbox must be set up on the CPU Configuration Setup menu, as described in chapter 10, *Expanded Functions Menu.* The default selection is "not enabled."

Contents of the computer mailbox registers are shown below:

Computer Mailbox

Bus Controller Address	Register 1
Operation (Read/Write)	Register 2
Communications Status	Register 3
I/O Address of Target	Register 4
Mailbox Address for Data	Register 5
Data Length in Bytes	Register 6
data registers Beginning address given in register 5, length in register 6.	to Register 70

SECTION 4
Reserved References

When creating a ladder logic program, be careful not to use either I/O or register references that are already assigned for use by system modules.

Required I/O References

Your program must reserve special I/O references for the following. (Parentheses indicate I/O references that can be changed.)

References	Type of Module	Function
(I0993-I1000)	Advanced I/O Receiver Module	Default status byte. (Can be changed.)
I1001-I1008	Interrupt Input Module	Status byte.
I1009-I1016	Communication Control	CCM Status byte.
I1009-I1016	LAN Interface Module	Status byte.
I1017-I1024	I/O Transmitter Module (Main I/O status table)	Status byte. Also at same relative location for each Expanded I/O channel.
I1024	Redundant Processor Unit	RPU status. Set to 1 if RPU is operating as backup CPU.
(O0993-O1000)	Advanced I/O Receiver Module	Default control byte. (Can be changed.)
O1015	Active Override Search	Enable bit (Expanded II only).
O1016	Active Override Search	Report bit (Expanded II only).
O1017	Redundant Processor Unit	Prevents CPU from being a master CPU, or switches to back-up unit.
O1022	Redundant Processor Unit	Transfers O0001 to O1016 from master to backup CPU.
O1023	Redundant Processor Unit	Transfers R254 from master to backup CPU.
O1024	Redundant Processor Unit	RPU control. Transfers registers whose range is specified by R255 (start) and R256 (end) from master to backup CPU.
(AI0993-AI1000)	Advanced I/O Receiver Module	Status byte. Also at same relative location for each Expanded I/O channel enabled. (Can be changed.)
AI1001-AI1008	Interrupt Input Module	Status byte.
AO0993-AO1000	Advanced I/O Receiver Module	Control byte. Also at same relative location for each Expanded I/O channel enabled.

Required Register References

Your program must reserve special register references for the following optional modules. (Parentheses show register references that can be changed.)

References	Type of Module	Function
R128	I/O Transmitter Module in Expanded mode	Auxiliary I/O status table diagnostic byte
(R240-R244)	ASCII/BASIC Module	RTU master command registers. Defaults, can be changed.
R247-R248	Communication Control Module	Software configuration of CCM (optional).
(R249-R253)	Data Processor Unit	Default status registers. Can be changed.
R254	Redundant Processor Unit	Status register. Content of R254 in backup CPU transferred to master or set to 0 if backup not present or available.
R255	Redundant Processor Unit	Control register. Content of R255 is first register for data transfer from master to backup CPU.
R256	Redundant Processor Unit	Control register. Content of R256 is last register for data transfer from master to backup CPU.
R256	I/O Transmitter Module in Expanded mode	Channel 1 diagnostic byte
R384	I/O Transmitter Module in Expanded mode	Channel 2 diagnostic byte
R512	I/O Transmitter Module in Expanded mode	Channel 3 diagnostic byte
R640	I/O Transmitter Module in Expanded mode	Channel 4 diagnostic byte
R768	I/O Transmitter Module in Expanded mode	Channel 5 diagnostic byte
R896	I/O Transmitter Module in Expanded mode	Channel 6 diagnostic byte
R1024	I/O Transmitter Module in Expanded mode	Channel 7 diagnostic byte
R1280	I/O Transmitter Module in Expanded mode	Channel 9 diagnostic byte
R1408	I/O Transmitter Module in Expanded mode	Channel A diagnostic byte
R1536	I/O Transmitter Module in Expanded mode	Channel B diagnostic byte
R1664	I/O Transmitter Module in Expanded mode	Channel C diagnostic byte
R1792	I/O Transmitter Module in Expanded mode	Channel D diagnostic byte
R1920	I/O Transmitter Module in Expanded mode	Channel E diagnostic byte
R2048	I/O Transmitter Module in Expanded mode	Channel F diagnostic byte
R400-R485	Loop Management Module	LMM Mailbox
R509, 759, 2009, 2259, 3009, 3259, 3509, 3759, 5009, 5259, 5509, 5759, 7009, 7249, 7509, 7759	Loop Management Module	Required register for LMM 1 to LMM 16.
R8060-R8074	I/O Interface Module for Series 90-70 I/O	Command block
R8075-R8122	I/O Interface Module for Series 90-70 I/O	Status. 3 registers per rack, up to 16 racks (for example, R8075-R8077 for rack 1)

Selectable References for Optional Modules

In addition to the required references listed previously, optional modules use these selectable I/O and register references. Quantities listed below are for one module. Multiply this number by the number of additional modules in the system for the total number of references needed.

References	Type of Module	Function
48 inputs	Bus Controller Module (with diagnostics)	First input corresponds to backplane address of module.
8 inputs	Bus Controller Module (without diagnostics	First input corresponds to backplane address of module.
8 inputs	I/O Communication Control Module	First input corresponds to backplane address of module.
8 inputs	I/O Link Local Module	First input corresponds to backplane address of module.
8 inputs	Loop Management Module	First input corresponds to backplane address of module.
8 inputs	ASCII/BASIC Module	First input corresponds to backplane address of module.
48 outputs	Bus Controller Module (with diagnostics)	First output corresponds to backplane address of module.
8 outputs	Bus Controller Module (without diagnostics)	First output corresponds to backplane address of module.
8 outputs	Loop Management Module	First output corresponds to backplane address of module.
8 outputs	ASCII/BASIC Module	First output corresponds to backplane address of module.
4 registers per I/O rack.	I/O Link Local Module	Table of 30 consecutive registers. First register corresponds to backplane address of module.
Up to 7 registers	Data Processor Unit	For each DPREQ used.
10 registers per LMM, 15 per loop	Loop Management Module	R0500-R7999 are used for data table in a ProLoop system.
6 registers	I/O Communication Control Module	First register corresponds to beginning I/O address of module.

Advanced I/O Receiver Module

The Advanced I/O Receiver Module can be used to provide diagnostic information about conventional I/O in the system. For more information about the I/O Receiver Module, refer to the data sheet (GEK-90771) accompanying this module.

The Advanced I/O Receiver Module is originally configured to respond as though it were a standard I/O Receiver card. To use the diagnostics features of the card, switch 3 of DIP switch pack U5 on the card must be set to the OPEN position. The diagnostics features that can be used depend on the settings of other switches on the card. Note that the board-edge LED indicators are active for either the OPEN or CLOSED position of switch 3. If you are not sure whether diagnostics is enabled, check the switch setting.

Setting the Diagnostics Enable switch to OPEN activates address selection for the card. Because diagnostics information must be transferred through the ladder logic, each Advanced I/O Receiver Module in the system must have an input address and output address assigned. Address assignments are made using the two 7-segment DIP switches on the lower left corner of the card.

The left set of switches sets the input address and the right set selects the output address. Selecting the addresses establishes the input and output status table addresses used by the module. These switches are factory-set for input and output references 0993 through 1000. However, these addresses may be changed for the application. The address is the same whether Normal or Expanded addressing is used. In an Expanded I/O system, the address is preceded by a channel number. That number is determined by the I/O Transmitter Module to which the I/O Receiver Module is connected.

The input byte may be used to detect I/O chain faults, input and output parity errors, and power supply problems, and to monitor the I/O Rack watchdog timer.

The output byte may be used to turn off all outputs downstream of the module, disable chain faults from stopping the Series Six CPU, disable downstream inputs from being reported into the CPU's Input state tables, and allow all downstream outputs to hold their last state in the event the Series Six CPU stops.

ASCII/BASIC Module

The ASCII/BASIC Module (ABM) is an intelligent communications module which also provides BASIC programming capabilities. Use of an ASCII/BASIC Module requires a CPU with the Advanced (Extended), Expanded, or Expanded II function set. For more information about the ABM, refer to the *ASCII/BASIC Module User's Manual* (GEK-25398).

I/O Addressing for the ASCII/BASIC Module

The ASCII/BASIC Module address is set by the DIP switches inside the backplane, to the right of the module.

The ASCII/BASIC Module uses 8 input references and 8 output references. These always map into the Main I/O status table. Therefore, in Expanded I/O systems, avoid addressing ASCII/BASIC and Loop Management Modules or IOCCM at the same relative address. Also, within the same channel, avoid overlapping any other I/O points at the ASCII/BASIC Module address.

Input references are called the Status Byte. Output references are called the Command Byte. The Status Byte contains information about the status of the module. For example: module OK, battery OK, module running, or powerup. The Control Byte may be used for generating interrupts in a running GEBASIC program, and may start or stop the module using ladder logic (rev C or later for 20K, rev D or later for 12K modules).

Including an ASCII/BASIC Module in the Ladder Logic Program

The program must include logic for communicating with the ASCII/BASIC Module.

Non-Expanded: For a Non-Expanded I/O system, use a DPREQ instruction. The value in the assigned DPREQ register reference is equal to the actual I/O address (the card address) of the ASCII/BASIC Module plus 1000. Refer to chapter 13, *Advanced Functions*, for instructions.

Expanded: For an Expanded I/O system using real Expanded I/O devices, do not use a DPREQ. Instead, use the Window function (part of the Expanded function set). Refer to chapter 10, *Expanded Functions Menu*, for instructions.

In GE BASIC, Expanded I/O references are accessed using read and write registers.

Bus Controller

A Bus Controller Module interfaces a Genius I/O serial bus to the Series Six PLC. The bus controller can be used with either Normal I/O addressing or Expanded I/O addressing. If the system uses Expanded I/O addressing, a DIP switch in the bus controller can be used to enable channel selection (instead of using an I/O Transmitter Module for channel selection). There are two types of bus controllers, bus controllers without diagnostics and bus controllers with diagnostics.

Genius Diagnostics data was discussed previously in this chapter. For more information about bus controllers in a Genius I/O system, refer to the *Genius I/O System User's Manual* (GEK-90486).

I/O Addressing for Bus Controllers

A bus controller without diagnostics requires one byte (8 references) of input address and 8 outputs.

Each bus controller with diagnostics requires 48 input and 48 output addresses, which are used for diagnostic input data. Of these, the first 32 are used for command output data. The rest are used by the CPU to report diagnostics. None of these references can be used for other purposes.

Actual addresses must be assigned on 64-bit boundaries if CPU diagnostics are to be entered in the Series Six Plus CPU. The inputs and outputs begin at the same relative address. Because the bus controller uses only the first 48 addresses, the final 16 addresses in each 64-address block can be assigned for general I/O use.

The bus controller address is set by the DIP switches inside the backplane, adjacent to the module. In an Expanded I/O system, the address is preceded by a channel number. Each bus controller must have a card address that is different from every other bus controller on that chain.

Including a Bus Controller in the Ladder Logic Program

No program logic is needed to enable a bus controller used with Non-Expanded addressing. If Genius diagnostics are used with Expanded addressing, you must enter the [CPU CONFIGURATION DATA] instruction at rung 1 and select the channels to be scanned in the CPU Configuration menu.

Communication Control Module (CCM)

The Communication Control Module (CCM) is an executive-level communication device that interfaces to the Series Six CPU. It is capable of communicating while the CPU is stopped or running. The CCM cannot be used in the same PLC system as the LAN Interface Module. For more information about the CCM, refer to the *Series Six Data Communications User's Manual* (GEK-25364).

CCM Status Byte

Eight input references, called the Status Byte, provide information on the module's operating status. The program may monitor these references to determine if there has been a communications error, if the port is busy, if an external device has communicated with the CCM, and if the CCM communications are operating properly. The CCM Status Byte is always located at address I1009 through I1016.

Port 1 and Port 2 Configuration Registers

Port 1 and port 2 of the Communications Control Module can be configured in software using registers R0247 and R0248. Configuration can be performed while a program is running in the CPU. To avoid inaccuracies, it is recommended that these two registers be reserved for the CCM if software configuration will be used.

Programming used for the Communication Control Module

The CCM may be configured for master/slave or peer-to-peer operation. When operated as a slave device, no program logic is needed to support the device. As a master device, or for peer-to-peer use, communication may be initiated by a Serial Communication Request (SCREQ) instruction in the program. The SCREQ function uses a register reference as the command to specify the type of data transfer that will occur. The next five register references are used to specify the parameters associated with the SCREQ function. Use of the SCREQ instruction is described in chapter 13, *Advanced Functions*.

In addition to communications initiated by SCREQ instructions, the CCM communicates with the CPU as part of the executive window. This communication occurs once each scan unless disabled in the program with a Status function. If the Status function is active, no communication between the CPU and the CCM will take place (either during the executive window, or SCREQ instructions). Use of the Status function is described in chapter 14, *Expanded Functions*.

Data Processor Unit

The Data Processor Unit (DPU) is an intelligent device used for file storage and retrieval, as well as for serial communication to external devices. The DPU uses Series Six PLC register memory for controlling the types of operations to be performed, as well as for monitoring the status of current operations. For more information about the DPU, refer to the *Data Processor Unit Manual* (GEK-25363).

DPU Status Registers

Registers R0249 through R0253 are the default DPU status registers. DPU status is updated in these registers during each DPU executive window and DPREQ. If the default registers are not appropriate for an application, they can be changed to any group of 5 registers from R0001 through R1024. This is done using the System Configuration Mode of the DPU, as described in the DPU manual.

Programming for the DPU

Communication with the DPU may be initiated by a Data Processing Request (DPREQ) instruction in the program. Each DPREQ instruction for a DPU requires a block of up to 7 registers. These must be located in the range of R0001 through R1024. Valid data transfers between the DPU and the CPU can only occur in this range of registers.

The first register in the block is the reference for the DPREQ. The value in the DPREQ register reference is a hexadecimal number from 1 to 64 (decimal 1 to 100). That number represents a DPU command, and specifies the type of data transfer that will occur - for example, a Read or a Write. Use of DPU commands is described in the DPU manual. The next six (or fewer) register references are used to specify the parameters for the data transfer.

Use of the DPREQ instruction is described in chapter 13, *Advanced Functions.*

In addition to communications initiated by DPREQ instructions, the CPU communicates with the Data Processor Unit as part of the executive window. This communication occurs once each scan unless disabled in the program with a Status function. If the Status function is active, no communication between the Series Six PLC and the DPU will take place (either during the executive window, or DPREQ instrunctions). Use of the Status function is described in chapter 14, *Expanded Functions.*

I/O Communication Control Module

The I/O Communication Control Module (IOCCM) is a communication device that is installed in the Series Six PLC parallel I/O system. It does not function in a Remote I/O chain. The IOCCM is similar in functionality to a CCM type 3. For more information about the module, refer to data sheet (GEK-90824) that accompanies this module.

The IOCCM has two ports, which are configured in hardware. No register references are needed for port configuration.

IOCCM Status Byte

Eight input references, called the Status Byte, provide information on the module's operating status. The eight inputs used for the Status Byte correspond to the module's card address, set on the backplane.

Addressing for the I/O Communications Control Module

The IOCCM may be installed at any address within the range 0001 through 0993. The card address is set by the DIP switches inside the backplane, adjacent to the module. The address is the same whether Normal or Expanded addressing is used. If multiple IOCCMs are used, they must be set to different addresses using the DIP switches.

IOCCM Command Register

The IOCCM uses six registers for initiating communications with an external device. The beginning address, called the command register, is the same as the beginning I/O address of the module. For example, if the card address of the module begins at 25 (and ends at 32), the command register for that module will be R0025. In this example, registers 0025 through 0030 will be used for the communications data. Note that the reference for the DPREQ register (see below) must not be the same as the command register reference.

Including an IOCCM in the Ladder Logic Program

In order to communicate with the Series Six CPU, the program must use an active Data Processor Request (DPREQ). The DPREQ register reference must not be the same as the command register. The value contained in the DPREQ register must be equal to the I/O address of the IOCCM plus 1000.

I/O Link Local Module

The I/O Link Local Module is used to interface the Series Six CPU to the I/O system of a Series One or Series Three PLC. For more information about the module, refer to the *I/O Link Local Module User's Manual* (GEK-90825).

Eight input references, called the Status Byte, provide information on the module's operating status. The eight inputs used for the Status Byte correspond to the module's card address, set on the backplane.

Addressing for the I/O Link Local Module

The I/O Link Local Module may be installed at any address within the range 0001 through 0993. The card address is set by the DIP switches inside the backplane, adjacent to the module. The address is the same whether Normal or Expanded addressing is used. In an Expanded I/O system, the address is preceded by a channel number. That number is determined by the I/O Transmitter card to which the module is connected. If multiple I/O Link Local Modules are used, they must be set to different addresses using the DIP switches.

Including an I/O Link Local Module in the Ladder Logic Program

In order for an I/O Link Local Module to communicate with the Series Six CPU, the program must provide an active Data Processor Request (DPREQ) function. The value in the register reference for the DPREQ must be equal to the address of the module plus 1000. No other I/O may be installed at the same address.

Setting Up the Configuration Table in the Program

In addition to a DPREQ, the program must include a configuration table for each I/O Link Local Module in the system. The configuration table is a block of registers that define the Remote Rack I/O and Series Six PLCI/O mapping. A group of four registers is used for each I/O rack in the system.

The module's inputs are used to indicate configuration errors and various types of communication errors within the remote I/O racks.

I/O Link Local Module: Configuration Registers	
Rn	Number of remote racks on port 2
Rn+1	Number of input points from rack ID 1
Rn+2	Number of output points to rack ID 1
Rn+3	Starting register for input table: rack ID 1
Rn+4	Starting register for output table: rack ID 1
Rn+5	Number of input points from rack ID 2
Rn+6	Number of output points to rack ID 2
Rn+7	Starting register for input table: rack ID 2
Rn+8	Starting register for output table: rack ID 2
.	
.	
Rn+26	Number of input points from rack ID 7
Rn+27	Number of output points to rack ID 7
Rn+28	Starting register for input table: rack ID 7
Rn+29	Starting register for output table: rack ID 7

The beginning register address (R) is determined by the DIP switch settings, as explained below. The length of the table depends on the number or racks in the system. One rack requires 5 registers and 7 racks require 29 registers.

Hardware Address for the Configuration Table

Two banks of DIP switches on an I/O Link Local Module must be set to specify the beginning register for the configuration table. All switches in Bank A and 6 switches in Bank B must be set. Below, x means a switch is in the OPEN position.

Starting Address	Bank A								Bank B								Bank C	
lsb	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2
R00001	x													x	x	x	x	x
R00002		x												x	x	x	x	x
R00003	x	x												x	x	x	x	x
R00004			x											x	x	x	x	x
R00005	x		x											x	x	x	x	x
.					x										x			x
.					x										x			x
R16379	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Interrupt Input Module

The Interrupt Input Module is used to respond very quickly to either ON-to-OFF or OFF-to-ON input changes. The desired response of each input is selected by a jumper on the card. The factory setting is for OFF-to-ON transitions. For more information about the Interrupt Input Module, refer to data sheet (GEK-83524) that accompanies this module.

Programming with the Interrupt Input Module

When a transition is detected, the program logic will jump to an interrupt subroutine at the end of the main program. An Interrupt Input Module located in the Main I/O chain will branch to one of the first 8 interrupt subroutines. An Interrupt Input Module located in the Auxiliary I/O chain will branch to one of the interrupt subroutines located in the second group of 8 subroutines. If the program includes other subroutines, the number available for Interrupt Input Modules will be limited, since the maximum number of subroutines in one program is 16.

I/O Mapping for the Interrupt Input Module

An Interrupt Input Module located anywhere in the Main I/O chain (Expanded or Non-Expanded), will always map its inputs to the Main I/O status table, from I1001 through I1008. Similarly, an Interrupt Input Module located anywhere in the Auxiliary I/O chain will always map into the Auxiliary I/O table at AI1001 to AI1008.

LAN Interface Module

The Local Area Network (LAN) Interface Module is a communications module that provides direct LAN attachment to an IEEE 802.4 carrierband network. It offers both Datagram and Global Data communications, as well as station configuration, managemant, and diagnostic tools. The LAN Interface Module cannot be used in the same PLC system as the CCM Module. If CCM capabilities are required, an IOCCM Module must be used instead. For more information about the LAN Interface Module, refer to the *GEnet Factory LAN Network Interface User's Manual* (GFK-0013).

Eight input references, called the Status Byte, provide information on the module's operating status. The program may monitor these references to determine if there has been a communications error, if the port is busy, if an external device has communicated with the LAN Interface Module, and if communications are occurring properly. The LAN Status Byte is always located at address I1009 through I1016.

Programming for the LAN Interface Module

Ladder logic programming for the LAN Interface Module can include:

1. Using the STATUS instruction to enable the communications window through which a request is transferred between the CPU and the LAN Interface Module.
2. Using the SCREQ instruction to open a window between the CPU and the LAN Interface Module. This window permits transfer of information in the Communication Control Block.
3. Programming the Communications Control Block, which consists of a block of registers that specify the command number and give the parameters of the command.
4. Using the LAN Status Byte to monitor the status of communications requests.

Loop Management Module

Proloop continuous process controllers interface to the Series Six PLC via one or more Loop Management Modules (LMM) on an RS422 serial link. An LMM installs in the Series Six PLC Local I/O system and acquires Proloop operating data which is stored in register memory. For more information about the Loop Management Module, refer to the *ProLoop Process Controllers System Manual* (GEK-90802).

Addressing for ProLoop

The Loop Management Module uses 8 inputs and 8 outputs of I/O. The 8 inputs are called the Status Byte, and contain information about module operation. The 8 outputs are called the Control Byte, and are used to start, stop, and initialize the LMM. These are always mapped into the Main I/O status tables. Avoid addressing other I/O at the same relative address.

Operating data for each control loop is stored in registers which are referenced according to the I/O address of the Loop Management Module.

A ProLoop system can include up to 16 Loop Management Modules. This is shown in the lefthand column of the table below. Each LMM can be assigned one of seven or eight available addresses. The possible addresses for each LMM are shown in the center of the table. The Loop Management address is set by the DIP switches inside the backplane, adjacent to the module.

LMM	Possible I/O Addresses for that LMM								Data Table	CDM
01	001	129	257	385	513	641	769	897	R0500-R0740	R0509
02	009	137	265	393	521	649	777	905	R0750-R0999	R0759
03	017	145	273	401	529	657	785	913	R2000-R2249	R2009
04	025	153	281	409	537	665	793	921	R2250-R2499	R2259
05	033	161	289	417	545	673	801	929	R3000-R3249	R3009
06	041	169	297	425	553	781	809	937	R3250-R3499	R3259
07	049	177	305	433	561	789	817	945	R3500-R3749	R3509
08	057	185	313	441	569	797	825	953	R3750-R3999	R3759
09	065	193	321	449	577	805	833	961	R5000-R5249	R5009
10	073	201	329	457	585	813	841	969	R5250-R5499	R5259
11	081	209	337	465	593	821	849	977	R5500-R5749	R5509
12	089	217	345	473	601	829	857	985	R5750-R5999	R5759
13	097	225	353	481	609	837	865	993	R7000-R7249	R7009
14	105	233	361	489	617	845	873	-	R7250-R7499	R7249
15	113	241	369	497	625	853	881	-	R7500-R7749	R7509
16	121	249	377	505	633	861	889	-	R7750-R7999	R7759

Each Loop Management Module uses 10 registers of memory. In addition, each loop used by that LMM uses 15 more registers. The range of registers reserved for each LMM are listed in the third column above.

NOTE

Registers R0400 through R0485 are used for the LMM mailbox.

Including ProLoop in the Ladder Logic Program

In order to communicate with the Series Six CPU, the program must include an active Data Processor Request (DPREQ) for each LMM in the system. The value in the assigned DPREQ register reference is equal to the actual I/O address of the Loop Management Module plus 1000. It is necessary to interlock the communications of LMMs and ASCII/BASIC Modules in the system to be sure that they do not attempt to use the same reference area at the same time.

The DPREQ may exhibit I/O conflicts when used in Expanded I/O systems. Therefore, the Window function (part of the Expanded function set), should be used instead.

Parallel I/O Transmitter Module

The Parallel I/O Transmitter Module extends the allowable distance for the Series Six PLC Local I/O chain to a total cable length of 500 feet (150 meters). Up to four parallel I/O transmitters may be installed in series, to extend the total length of the local parallel chain to 2000 feet (600 meters). For more information about the I/O Transmitter Module, refer to data sheet (GEK-83515).

The Parallel I/O Transmitter card comes in two versions. The enhanced model (IC600BF900C or later) supports Expanded I/O systems. It multiplexes the parallel I/O chain. The enhanced model also provides additional diagnostic information via 8 inputs.

Selection of Normal or Expanded I/O Addressing

The Enhanced I/O Transmitter Module can be used to enable Expanded I/O addressing (the bus controller can also be used to enable Expanded I/O addressing). In Normal addressing mode, there is one Main I/O chain and one Auxiliary I/O chain (optional). In Expanded I/O addressing, there are up to 16 I/O chains. Each one is referred to as a channel. If more channels are used in addition to the Main and Auxiliary I/O status tables, each channel must be driven by an I/O Transmitter Module or bus controller. The addressing mode of the enhanced I/O Transmitter Module is set by jumper J2 on the card.

Channel Selection for Expanded Addressing

If Expanded addressing mode has been selected (jumper J2 over pins 2 and 3), the channel number for the card is selected using the first three switches on the backplane DIP switch pack. DIP switch positions for each channel of I/O are shown below. In the chart, x means the switch is in the open position (closed to the left).

Channel Number		DIP Switch Position		
Main Chain	**Aux. Chain**	**3**	**2**	**1**
Main I/O table	Aux. I/O table			
1	9			x
2	A		x	
3	B		x	x
4	C	x		
5	D	x		x
6	E	x	x	
7	F	x	x	x

Program References for the I/O Transmitter Module

If the Expanded mode is enabled, the module uses the references listed below to report I/O channel status to the CPU. If Expanded mode is disabled, the module does not use these register references.

Channel Return Status Memory Locations

Main I/O Chain		Auxiliary I/O Chain	
Channel	**Location**	**Channel**	**Location**
Main I/O table	I1017-I1024	Aux. I/O table	R128 (high byte)
1	R256 (high byte)	9	R1280
2	R384	A	R1408
3	R512	B	R1536
4	R640	C	R1664
5	R768	D	R1792
6	R896	E	R1920
7	R1024	F	R2048

Expanded Addressing for a System with an RPU

Formerly, the Redundant Processor Unit used input I1024, outputs O1017, O1022, O1023, and O1024, and registers R254-256. Referring to the list above, you can see that there is a conflict with the channel status return memory locations for the Main I/O chain and channel 1. This conflict causes information from the I/O Transmitter Module channel being written into references assigned to the RPU. To avoid this conflict, a system which uses Expanded I/O addressing with I/O Transmitter Modules reporting status information should avoid using addresses 977 and up in channel 1 with Genius I/O, or avoid using the I/O Transmitter Module in channel mode for channel 1.

Redundant Processor Unit

The Redundant Processor Unit (RPU) monitors the CPU and I/O. When the RPU detects a failure of the CPU or I/O, it switches to a backup CPU and (optionally) to a backup I/O chain. For more information about the module, refer to the *Redundant Processor User's Manual* (GEK-25366).

I/O Addressing for the RPU

Originally, the RPU used pre-assigned inputs and outputs in the Main I/O status tables. These inputs and outputs could not be changed. The upgraded version of the RPU uses hardware-selectable I/O addresses.

The older type of RPU should not be used in a system with Expanded I/O addressing. The older RPU uses 8 outputs (O1017 through O1024) to control its operations. Setting O1017 in the master will cause transfer to the back-up CPU (if it is available). If O1022 is set, outputs from the Main I/O Status tables and the Override table are transferred from the master to the back-up. If O1023 is set, register R0254 is transferred from the back-up to the main CPU. If O1024 is set, a block of registers starting at the address specified by the value in register R0255, and ending at the value in register R0256 will be transferred from the master to the back-up. The 8 inputs (I1017 through I1024) and 8 outputs (O1017 through O1024) are always mapped into the Main I/O status table. Of the 8 inputs, only input 1024 is used. It is set to "1" if the associated CPU is the backup.

Only the upgraded model RPU should be used in an Expanded I/O system. In Expanded addressing, an I/O Transmitter Module in either channel 0 or channel 1 will write data into the references formerly assigned only to the RPU. For example, for channel 0, the status of the I/O Transmitter module maps into Main I/O at addresses I1017 through I1024. However, I1024 is the input normally used by the RPU to specify the end of the register block to be transferred from the main CPU to the backup CPU. The upgraded model RPU features jumper-selectable addressing.

NOTE

When using the RPU in channelized systems, do not use channel 1 because there is no way to avoid the conflict of register transfer for the backup CPU.

Series 90-70 I/O

Series 90-70 I/O modules are low-power VME bus-based I/O modules. These high-density I/O modules are smaller in size than conventional Series Six PLC I/O modules, and require Series 90-70 I/O racks and power supplies. Series 90-70 I/O can be used as the only type of I/O for the Series Six PLC. It can also be used in a system that includes standard Series Six I/O PLC modules and Genius I/O blocks. *To use Series 90-70 I/O, the PLC system must have a minimum register memory size of 8K.* If the Series Six PLC system includes Series 90-70 I/O, I/O and register reference use must be carefully planned, as discussed below. For more information about using Series 90-70 I/O with the Series Six PLC, refer to the *Series 90-70 I/O Application Guide* (GFK-0152).

Command Block for Series 90-70 I/O

The Series 90-70 I/O Command Block location in the CPU always begins at R8060, extending up to R8074. These references must always be reserved when the Series Six PLC system includes any Series 90-70 I/O racks. Registers in the command block have the following assigments:

R8060	=	Window access operand
R8061	=	Command number
R8062 to R8074	=	Command parameters

Reserved References for Series 90-70 I/O Racks

In addition to the Command Block reserved registers, I/O and register references must be reserved for the Series 90-70 I/O racks and modules in the system. The locations of these references depend on the number assigned to each rack. This is shown in the table below.

Rack Number	Series Six I/O Range	Window Address I/O Range	Rack Status
0	I/O 0001 - 0256	513 - 520	R8075 - R8077
1	I/O 0257 - 0512	521 - 528	R8078 - R8080
2	I/O 0513 - 0768	529 - 536	R8081 - R8083
3	I/O 0769 - 1000*	537 - 544	R8084 - R8086
4	I/O 0001 - 0256	545 - 552	R8087 - R8089
5	I/O 0257 - 0512	553 - 560	R8090 - R8092
6	I/O 0513 - 0768	561 - 568	R8093 - R8095
7	I/O 0769 - 1000*	569 - 576	R8096 - R8098
8	AI/AO 0001 - 0256	577 - 584	R8099 - R8101
9	AI/AO 0257 - 0512	585 - 592	R8102 - R8104
10	AI/AO 0513 - 0768	593 - 600	R8105 - R8107
11	AI/AO 0769 - 1000*	601 - 608	R8108 - R8110
12	AI/AO 0001 - 0256	609 - 616	R8111 - R8113
13	AI/AO 0257 - 0512	617 - 624	R8114 - R8116
14	AI/AO 0513 - 0768	625 - 632	R8117 - R8119
15	AI/AO 0769 - 1000*	633 - 640	R8120 - R8122

* To avoid addressing conflict, slot 9 in racks 3, 7, 11, and 15 should not be used for an I/O module.

Rack Number: The rack number (0 through 15) is assigned using jumpers on the rack backplane. Setting the rack number determines the I/O and register references that must be reserved for that rack, as shown in the preceding table.

I/O Status Table Addresses: The input and output modules in each Series 90-70 rack use up to 256 assigned I/O point addresses. The rack number and module location determine the locations of these addresses. Each rack has 8 slots for I/O modules and each slot is assigned 32 addresses (I/O points). When these addresses are used for Series 90-70 points, they must not also be used for Series Six I/O or Genius I/O points in the system. If I/O point ranges were overlapped, the Series 90-70 I/O Interface Module would overwrite the other device's I/O data.

Notice in the preceding table that the same I/O point addresses are assigned to every fourth rack. For example, Rack 0 and Rack 4 I/O module references are the same since they are mapped into the same relative addresses (I/O 0001-0256). Rack 0 and Rack 4 are called "complementary" racks. That means each input or output module in rack 0 is the complement of the module in the same slot in rack 4. For example, if slot 2 has an input module in rack 0, slot 2 in rack 4 can only have an output module. Therefore, rack 0 would use the input addresses assigned to slot 2, while rack 4 would use the output addresses assigned to slot 2. Using complementary modules prevents I/O addressing conflicts.

CAUTION

I/O addresses not actually used for Series 90-70 I/O points could be assigned with care to conventional Series Six PLC I/O points.

Window Address I/O Range: The beginning address used by the rack to communicate with the CPU. This is similar to setting the communications address of a conventional Series Six PLC I/O module with the DIP switches on the rack backplane. Each Series 90-70 rack uses eight assigned I/O addresses for communications. When a communications address is used for a Series 90-70 module, it must not be assigned to a Series Six PLC I/O module or Genius I/O block in the system. Assigning the same communications address to a conventional I/O module or a Genius I/O block and a Series 90-70 I/O module could result in bus interference if the conflicting modules attempted to communicate with the CPU at the same time.

Rack Status Block Location: Series 90-70 I/O uses CPU registers R8075 through R8122 to report the status of I/O communications and block services requested. Each I/O rack uses a status block of three registers.

Register 1 - Window Completion Status: Shows the status of the communications window.

Register 2 - Module Active Register: Bits in this register indicate the activity of input and/or output modules in the rack. The bits from left to right correspond to slots in the rack. Each time the rack's I/O is scanned, bits are set for each module that communicates successfully.

Register 3 - Module Present Status and Module Status Register: The low byte of the third register indicates if the module is present as determined by a read of the board ID data. The high byte of the third register is used for status data from the module.

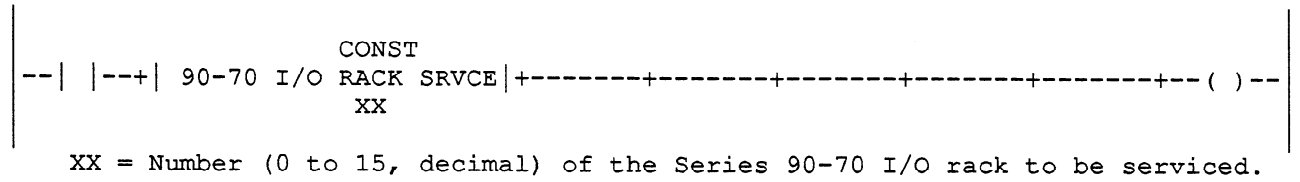
Servicing Series 90-70 I/O

Series 90-70 I/O service is not part of the regular I/O service portion of the CPU Sweep. Instead, each Series 90-70 rack is serviced during execution of the ladder logic. This allows synchronous data interchange between the CPU and the Series 90-70 I/O modules. Time required for each rack is typically between 3ms and 4ms, depending on the rack configuration and transmission distance between the rack and the CPU.

Since the Series 90-70 I/O is not serviced automatically by the CPU, the ladder logic program must include instructions to the CPU to service the Series 90-70 I/O racks. These instructions provide commands to the Series 90-70 I/O Interface Module. When the command is received, the I/O Interface Module executes the command and returns the I/O state values to the CPU immediately.

Using the Expanded II Instruction Set

The Expanded II instruction set, in combination with Logicmaster 6 release 4 software, provides a special instruction to service a Series 90-70 rack. All that is needed to program this instruction is to enter the rack number of the I/O rack to be serviced. All I/O modules in that rack will be updated when the instruction is executed in the ladder logic program.



For more information on Series 90-70 I/O ladder logic programming, see the *Series 90-70 Application Manual* (GFK-0152).

Chapter 13
Advanced Functions

GEK-25379

The Advanced Function Set includes most of the Series Six PLC programming instructions:

- Relays
- Timers and Counters
- Shift/Move Functions
- Basic Arithmetic Functions
- Special Functions
- Data Move Functions
- Signed Arithmetic Functions
- Table Move Functions
- List Functions
- Matrix Functions
- Control Functions

Chapter 5, *Edit Program*, explained how to perform the editing involved in creating a program. Chapters 13, *Advanced Functions*, and 14, *Expanded Functions*, define the functions that can be included in a program.

Use this chapter as a reference to the Advanced functions. Within the chapter, functions are grouped into sections that correspond to the function key assignments used during programming. Within a section, each function is explained separately. After a description of the way the function operates in a program, you will find step-by-step instructions for entering the function in a rung.

Using the Advanced Function Set

The I/O and register references that can be entered for Advanced functions depend on entries set up in the Scratch Pad for CPU function level and register memory. In addition, some basic functions are limited in range. The descriptions in this chapter will specify the range of reference values you can enter.

If the CPU has an older function level, it may not be able to use all of the Advanced functions. For more information about CPU function level, see chapter 3, *Scratch Pad.*

To assure you of creating a program that is compatible with your Series Six PLC CPU, the Logicmaster 6 software restricts the use of program functions and register memory to the level set up in the Scratch Pad.

Configuring the Scratch Pad

Before programming, you should configure the Scratch Pad display screen to match the capabilities of the CPU. In the Scratch Pad, specify the CPU function set. It may be Basic or Extended (for older CPUs), Advanced, Expanded, or Expanded II. Also specify the amount of CPU register memory that is available.

SECTION 1
Relays

This section is a reference to the Relay functions:

- Normally Open Contact
- Normally Closed Contact
- Latch Coil
- One-Shot Coil
- Relay Coil

Normally Open Contact

For every scan that power is received, a normally open contact will close, passing power flow to the right. When it is open, its status is off and power is not passed to the right. A normally open contact can be assigned any available Main, Auxiliary, or Expanded I/O reference.

Entering a Normally Open Contact

A normally open contact can be placed anywhere in the first 9 columns and 8 lines of a rung. (Only 7 lines of a rung may be displayed on the screen at one time.) If more than nine contacts are needed in series, enter the first nine, and assign an output coil reference to a coil in position 10. Use a contact having this output coil reference as the first contact in the next rung of logic. In this way, as many contacts as needed can be placed in series.

1. Enter any logic required to control power flow to the contact.
2. With the cursor at the location for the contact, select Relay (F1) and then Normally Open (F1).

```
*******
---|  |---
```

3. Using the numeric keypad, enter a Main, Auxiliary, or Expanded I/O reference for the contact. After entering the reference, press CTRL-E (or the Enter key).
4. Complete the logic for the rung.

Normally Closed Contact

Every scan that power is received, a normally closed contact passes power flow to the right if it is closed (status is off). If it is open (status is on), power is not passed to the right. A normally closed contact can be assigned any available Main, Auxiliary, or Expanded I/O reference.

Entering a Normally Closed Contact

A normally closed contact can be placed anywhere in the first 9 columns and 8 lines of a rung. (Only 7 lines of a rung may be displayed on the screen at one time.) To enter more than nine contacts in series, enter the first nine and assign an output coil reference to a coil in position 10. Use a contact having this output coil reference as the first contact in the next rung of logic. In this way, as many contacts as needed can be placed in series.

1. Enter any logic required to control power flow to the contact.
2. With the cursor at the location for the contact, select Relay (F1) and then Normally Closed (F2).

```
 ******
---|/|---
```

3. Using the numeric keypad, enter a Main, Auxiliary, or Expanded I/O reference for the contact. After entering the reference, press CTRL-E (or the Enter key).
4. Complete the logic for the rung.

Relay Coil

A relay coil represents a simple output device that is on when power is received. Coil references can be used as often as necessary in a program. It is the last state of the coil that is transmitted to the output hardware at the end of the scan. Coils O1001 to O1024 are internal coils, not tied to hardware outputs.

Entering a Relay Coil

Coils can only be placed in the tenth column of the first line of a rung.

1. Enter the first line of logic, which ends at the coil. Additional lines of logic may also be entered now.
2. Select Relay (F7). The coil is displayed in the tenth column.

```
 *******
---( )
```

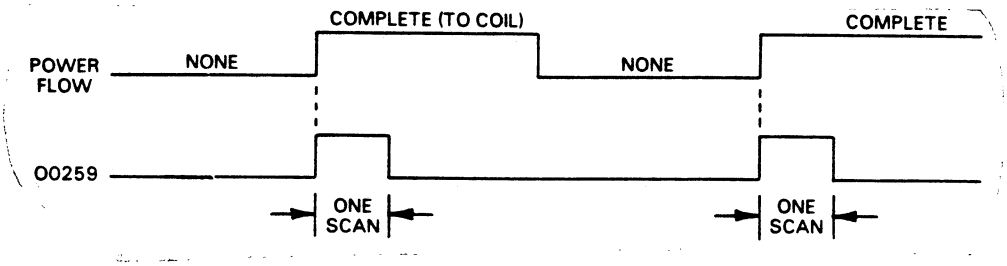
3. Using the numeric keypad, type in the reference for the contact. It may be any Main, Auxiliary, or Expanded output. After entering the reference, press CTRL-E (or the Enter key).
4. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

One-Shot Coil

A one-shot coil is energized in the same manner as a relay coil. However, once set it stays on only for one scan of the CPU. Power flow must be removed and then restored to generate a second short pulse. One-shot coils are activated by the off-to-on transition of the power flow resulting from the entire relay network, not just a single contact. This function is useful for controlling operations having a short duration, such as start commands or data sampling. A one-shot coil can only be used in the Main or Auxiliary Output status table.

The coil is on from the first time it is energized until it appears again on the following scan, regardless of the amount of time power flow is received. It is on for one complete scan from when it was energized to the end of that scan and then entering the next rung down, until the one-shot reference appears again. At that point, it turns off.

pcs6840269



Entering a One-Shot Coil

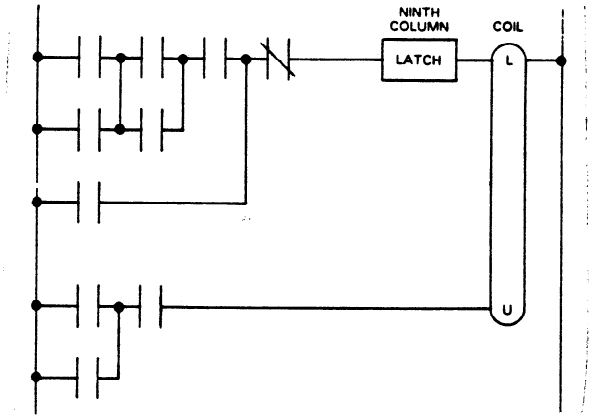
1. Enter the first line of logic, which ends at the coil. Additional lines of logic may also be entered now.
2. Select Coil (F7) and then One-Shot (F6). The one-shot coil is displayed in the tenth column.

```
*******
---(OS)
```

3. Using the numeric keypad, type in a Main or Auxiliary Output reference for the contact. Then, press CTRL-E (or the Enter key).
4. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Latch

The Latch function uses two signals to control the same coil at the end of a rung.



pcs6840268

When power flows to the coil from the first line of logic, the coil is turned on. This "sets" the latch. When power flows to the coil from the second line of logic, the coil is turned off. This "resets" or "unsets" the latch. If both lines supply power to the coil at the same time, the latch reset dominates and the coil is turned off.

Entering a Latch

A latch must be placed in the ninth column of the upper line of logic. Parallel logic may be used, with a total of 8 lines in the rung. (Only 7 lines of a rung may be displayed on the screen at one time.) There may only be two lines in the ninth column.

1. Enter the logic for the rung, leading to the latch in column 9. (See the note below.)
2. Select Coil (F7) and then Latch (F6). The latch is displayed in the ninth and tenth columns.

```
          *******
---|LATCH|---( L)
               (  )
               (  )
-------------(UL)
```

3. Using the numeric keypad, enter a Main, Auxiliary, or Expanded output for the coil. After entering the reference, press CTRL-E (or the Enter key).
4. When the rung is complete, press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

NOTE

For the latch rung to be valid, there must be logic in the second line. Remember that a program function (for example, a math function or a data move) only executes when it receives power flow *from the preceding function.* If you locate that program function at the left rail, it will only execute when the latch passes power flow. However, if you precede it with a contact at the left rail, the program function will execute whenever the contact passes power flow. The contact, which is next to the left rail, will operate correctly regardless of output power flow from the latch line.

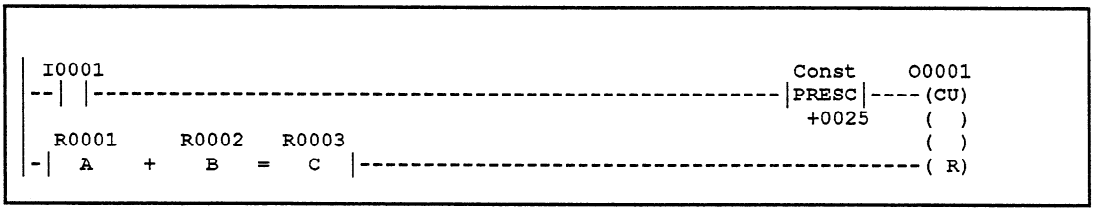
SECTION 2
Timers and Counters

This section is a reference to the Timer and Counter functions:

- Up Counter
- Down Counter
- Seconds Timer
- Tenths Timer
- Hundredths Timer

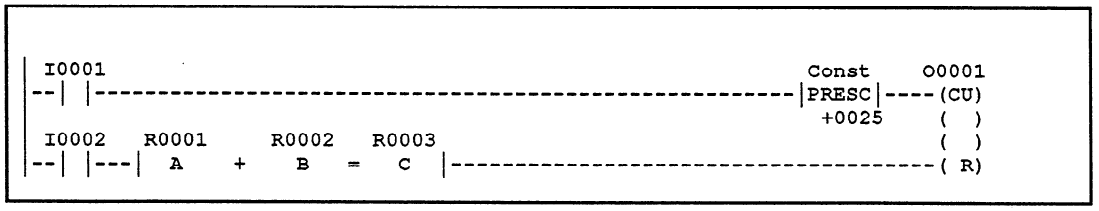
Using Logic in the Reset Rung of a Timer or Counter

When using logic in the second line of a timer or counter rung, remember that a program function (for example, a math function or a data move) only executes when it receives power flow *from the preceding function.* If you locate that program function at the left rail, it will only execute when the timer or counter in the first line of the rung passes power flow. For example:



In this example, the Addition function, which controls power flow to the reset, will only execute when the counter passes power flow.

However, a contact placed at the left rail will execute on every sweep (unless controlled by an MCR or a Skip function). Entering a contact at the left rail will cause the program function to execute on every sweep. The same example would look like this:



Now, the Addition function, which controls power flow to the reset, will execute every time the normally open contact passes power flow.

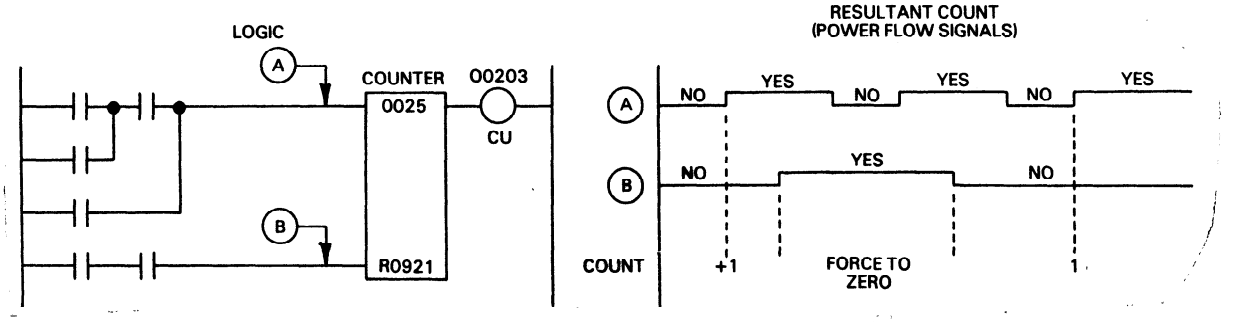
Counters

There are two types of counters, up counters and down counters. Both store the maximum value of the counter in a preset. Both also store the current value of the counter in an accumulate register. Counters increment or decrement upon detecting a transition in power flow from off to on. Detection of this transition is automatic.

Up Counters

An up counter operates like a timer. The bottom line resets the up counter to zero, the coil energizes when the preset is reached, and the count continues beyond the preset value until it reaches 65,535, the capacity of one register. The current value is incremented by one only by an off-to-on transition of the top signal. Power flow must be removed and reapplied to cause another increment. This illustration shows the power flow signals resulting from an up counter.

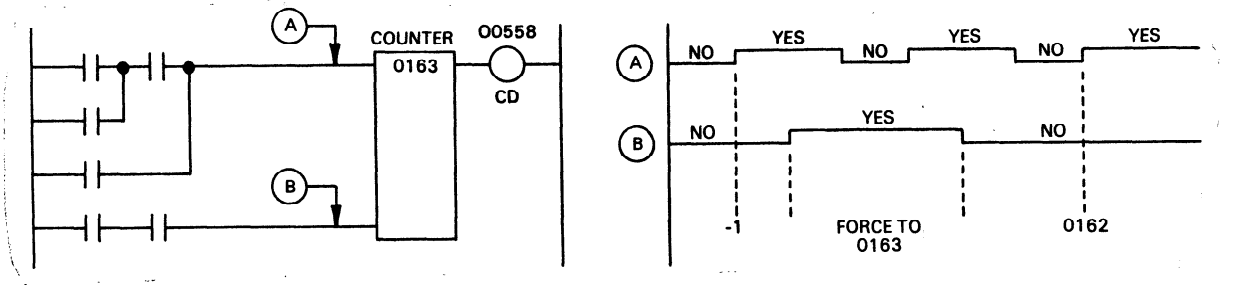
a40131



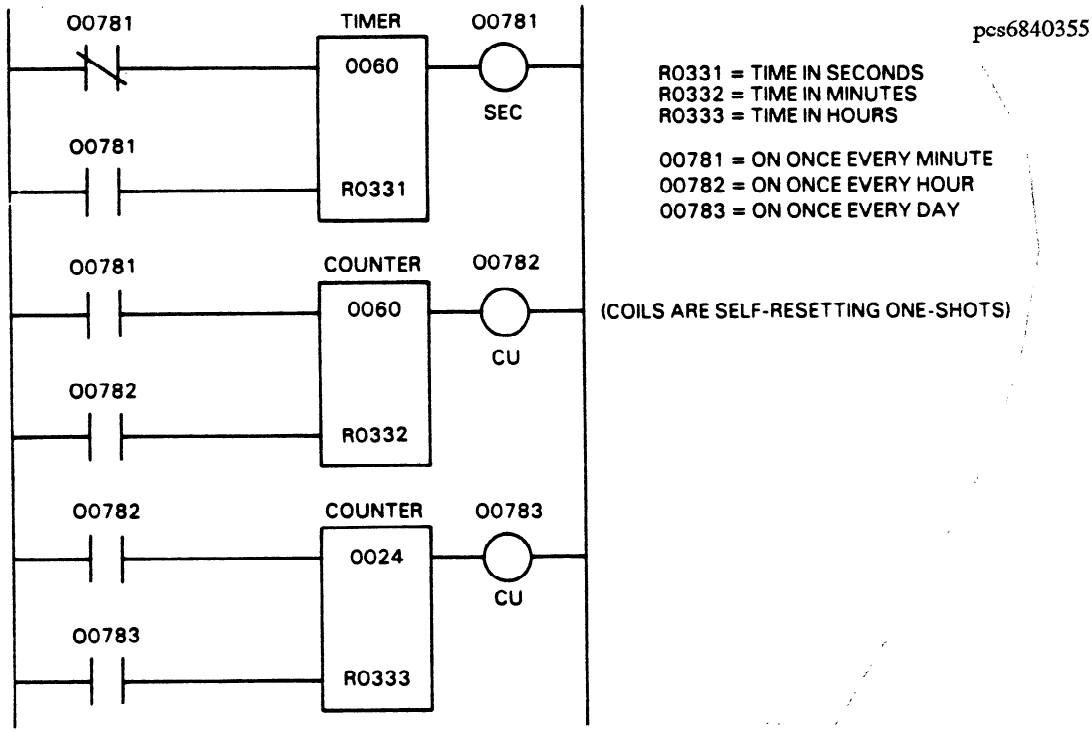
Down Counters

A down counter operates differently than an up counter or a timer. When continuity is received through the bottom line, the down counter resets the accumulate value to equal the preset value. Every time the top line receives power flow to the preset reference, the value in the register is decremented by one. At the value zero, the coil is energized and no further counts are recorded. Power flow must be removed and then reapplied to generate another decrement. This illustration shows the power flow signals resulting from a down counter.

a41032



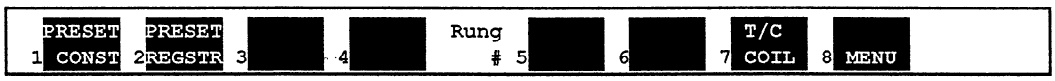
This illustration below shows a sample program using a timer and two counters programmed as a 24-hour clock.



Entering a Counter

A counter must be placed in the ninth column of the upper line of logic. Parallel logic may be used, with a total of 8 lines in the rung. (Only 7 lines of a rung may be displayed on the screen at one time.) There may only be two lines in the ninth column.

1. Enter the logic for the upper part of the rung, leading to the preset in column 9.
2. Select Timer/Counter (F2) and then Preset (F1). The cursor moves to the ninth column, and the screen displays these function key assignments:



3. To enter the preset, select either Preset Constant (F1) or Preset Register (F2). Press CTRL-E (or the Enter key). For a Preset Register, go immediately to step 4.

   A. If a preset constant is selected, the following appears at the cursor position:

```
    Const
---|PRESC|-
     ***
```

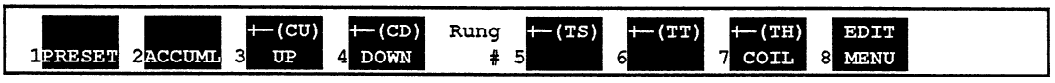
   B. Using the numeric keypad, enter the constant up to 999 for the preset. After entering the reference, press CTRL-E (or the Enter key). Go immediately to step 5.

4. To enter a reference for the preset, select Preset Register (F2). Then, press CTRL-E (or the Enter key). The following display appears at the cursor position:

```
   R*****
---|PRERG|-
      ***
```

   A. Enter any available register reference. After entering the reference, press CTRL-E (or the Enter key).

5. After the preset is entered, the cursor moves to column 10. The following function key assignments are active:



6. Select the type of counter by pressing the appropriate function key. The counter coil display appears in column 10. For example, this is the display for an up counter:

```
 *****
--(CU)
  (  )
  (  )
--( R)
```

7. Enter the reference for the counter. It may be any available Main or Auxiliary output. After entering the reference, press CTRL-E (or the Enter key).

8. If you have not already done so, enter the logic to control resetting the accumulate register now. When this logic is complete, select Accumulate (F2) to enter the Accumulate register. The Accumulate display appears in column 9:

```
   R*****
---|ACCRG|-
      ***
```

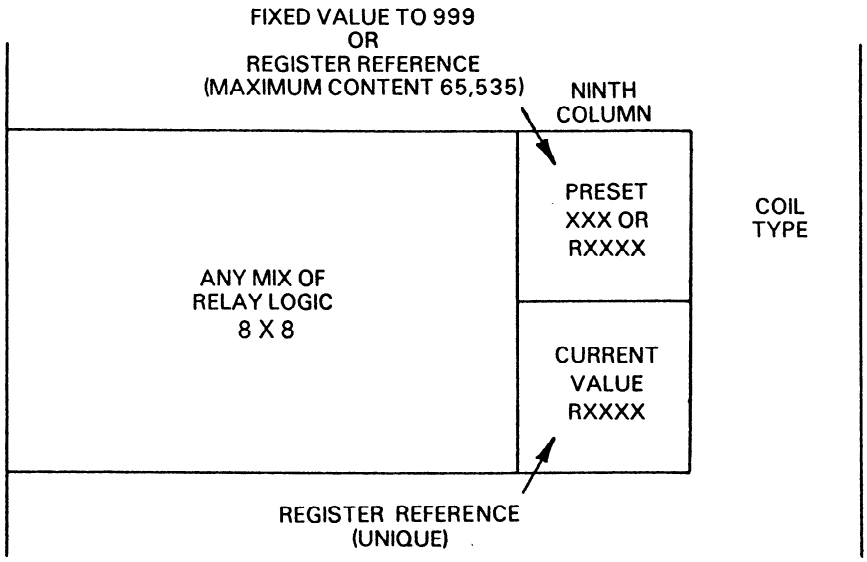
   Enter any available register. Then, press CTRL-E (or the Enter key).

9. When the rung is complete, press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Timers

A timer uses two signals to measure a specified amount of time. There are three types of timers: seconds, tenths of seconds (1/10 S or TT), and hundredths of seconds (1/100 S or TH).

pcs6840272



The upper part of the logic for the rung leads to the preset, which may be a constant or a register. The preset determines the maximum value for the timer. The maximum value of a constant preset is 999 or 99.9 or 9.99 seconds, depending on the timer increment. If a register is used as the reference for a timer, the maximum value it may contain is 65,535.

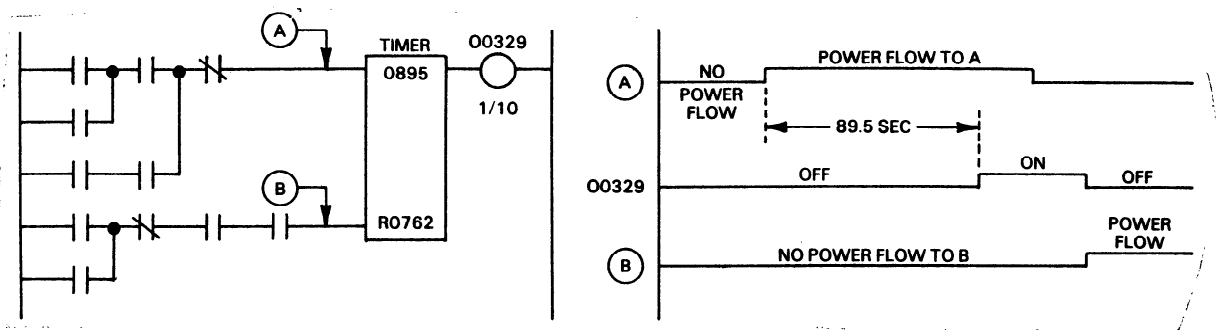
When power flows only through the top line, the timer is enabled and begins to record time in the selected increments. If power flow to the top of the rung is interrupted, the timer stops; but the value stored in its accumulate register is not reset to zero. When power flow is restored, the timer resumes at its stored value. When the value in the accumulate register equals or exceeds the preset value, the coil turns on.

The lower part of the logic for the rung (lines 2 through 8) leads to the register that accumulates the current value of the timer. When power flows to the bottom line of a timer, the value in the accumulate register is reset to zero, the coil is de-energized and no time is recorded.

All timers are based on the crystal in the CPU. Long-term accuracy is ±0.01 percent. Short-term accuracy is minus the time increment plus the scan time.

The following illustration shows the power flow signals resulting from a typical timer.

a41033

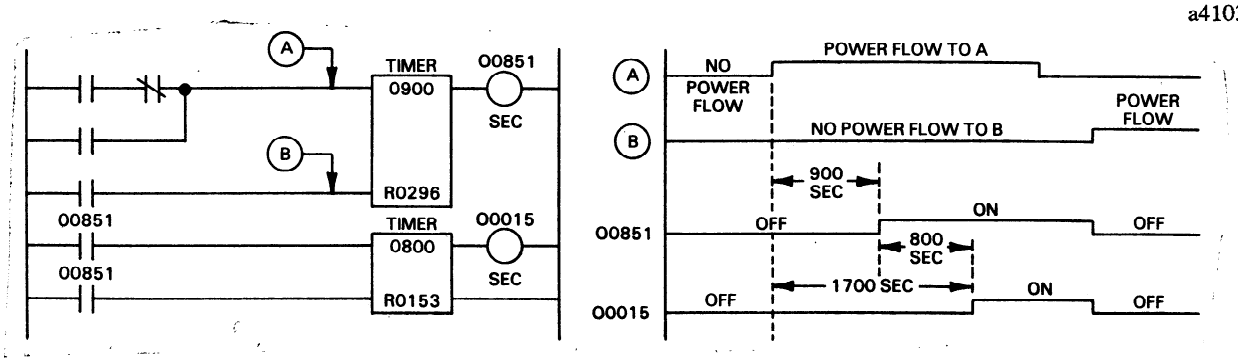


The timer coil is energized when the current time value equals or exceeds the preset value. Fixed presets have values up to 999. Register presets can be used for values up to 65,535 (18.9 hours). A timer operates after its coil is energized, up to the maximum value of 65,535, then stops.

Cascaded Timer

If longer periods must be timed, timers can be cascaded to increase their range while maintaining their accuracy.

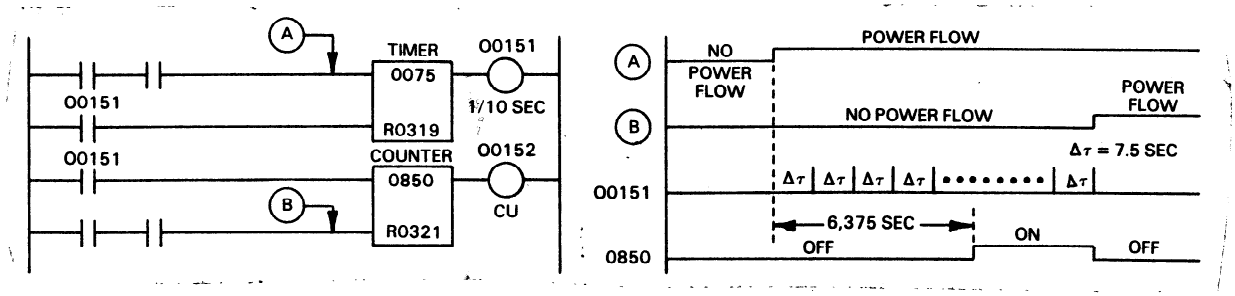
a41034



Master Timer

Greater range can also be obtained by the use of a master timer. A master timer establishes the basic accuracy, and counters are slaved from it. This method is less accurate; errors are accumulative.

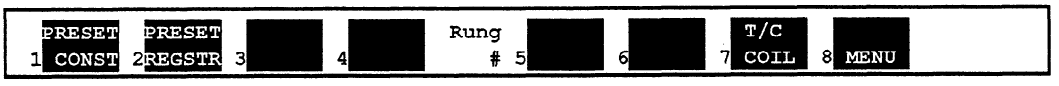
a41035



Entering a Timer

A timer must be placed in the ninth column of the upper line of logic. Parallel logic may be used, with a total of 8 lines in the rung. (Only 7 lines of a rung may be displayed on the screen at one time.) There may only be two lines in the ninth column.

1. Enter the logic for the upper part of the rung, leading to the preset in column 9.
2. Select Timer/Counter (F2) and then Preset (F1). The cursor moves to the ninth column, and the following function key assignments become active:



3. To enter the preset, select either Preset Constant (F1) or Preset Register (F2). Press CTRL-E (or the Enter key). For a Preset Register, go immediately to step 4.

   A. If Preset Constant is selected, the following display appears at the cursor position:

```
    Const
---|PRESC|-
     ***
```

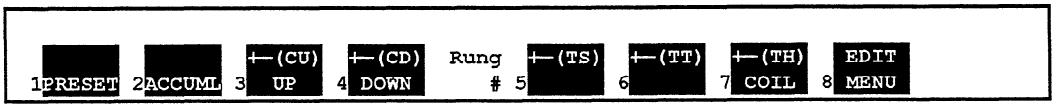
   B. Using the numeric keypad, enter a constant up to 999 for the preset. After entering the reference, press CTRL-E (or the Enter key). Go immediately to step 5.

4. To enter a reference for the preset, select Preset Register (F2). Press CTRL-E (or the Enter key). The following appears at the cursor position:.

```
   R*****
---|PRERG|-
     ***
```

   Enter any available register reference. Then, press CTRL-E (or the Enter key). Go immediately to step 5.

5. After the preset is entered, the cursor moves to column 10 and the following function key assignments are displayed:



6. Select the type of timer by pressing the appropriate function key. The timer coil display appears in column 10. For example, this is the display for a seconds timer:

```
 *******
--(TS)
  (   )
  (   )
--( R)
```

7. Enter the reference for the coil. It may be any available Main, Auxiliary, or Expanded I/O reference. After entering the reference, press CTRL-E (or the Enter key).
8. If you have not already done so, enter the logic to control resetting the accumulate register now. When this logic is complete, select Accumulate (F2) to enter the Accumulate register. The Accumulate display appears in column 9:

```
   R*****
---|ACCRG|-
     ***
```

   Enter the reference, which may be any available register. Then, press CTRL-E (or the Enter key).
9. When the rung is complete, press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

SECTION 3
Shift/Move Functions

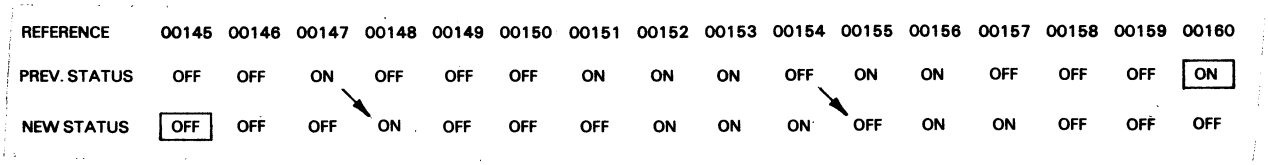
This section is a reference to the Shift/Move functions:

- Shift I/O
- I/O to Register Move
- Register to I/O Move
- Binary to BCD Conversion
- BCD to Binary Conversion

Shift I/O

The Shift I/O function shifts the values in 16 I/O locations one position upward. Every scan that power is received, the Shift I/O function copies the values in 16 adjacent I/O locations into the next higher address. For example:

pcs6840273



After the shift occurs, the lowest bit address is loaded with a 0. If an input address is specified, it is usually in an area where there are no hardware inputs for 16 consecutive references. The 16 locations whose values will be shifted must lie on a byte boundary (i.e., 0001, 0009, 0017). If an output address is specified, it may consist of either internal or hardware references.

The Shift I/O function generates power flow if the status shifted out of the high order reference is a 1 (ON).

Entering a Shift I/O Function

A Shift I/O function can be placed in columns 1 to 9 of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Shift/Move (F3) and then Shift I/O (F1).

```
  *****
-|SHIFT|-
```

3. The cursor is at SHIFT. Using the numeric keypad, type in the beginning address of the data to be shifted. It may be any Main I/O status table reference beginning on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
4. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

I/O to Register Move

The I/O to Register Move function copies the contents of 16 adjacent I/O references to a register.

Every scan that power flows to the function, the data is transferred. Power flow is generated whenever the function is active.

Entering an I/O to Register Move Function

An I/O to Register Move function can be placed in columns 1 to 8 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Shift/Move (F3) and then I/O to Register Move (F3).

```
  *****   R****
-| I/O  TO  REG  |-
```

3. The cursor is at I/O. Using the numeric keypad, type in the starting address for the source of the data. It may be any Main I/O status table reference beginning on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to TO REG. Enter a register to R1024 as the destination of the data. After entering the reference, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Register to I/O Move

The Register to I/O Move function copies the contents of a register to 16 adjacent I/O references.

Every scan that power flows to the function, the data is transferred. Power flow is generated whenever the function is active.

Entering a Register to I/O Move Function

A Register to I/O Move function can be placed in columns 1 to 8 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Shift/Move (F3) and then Register to I/O Move (F4).

```
  R****   *****
-| REG  TO  I/O  |-
```

3. The cursor is at REG. Using the numeric keypad, enter a register up to R1024 as the source of the data. Then, press CTRL-E (or the Enter key).
4. The cursor moves to TO I/O. Enter a reference in the Main I/O Status table as the starting address of the destination of the data. Then, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Convert Binary to BCD

The Binary to BCD conversion takes binary data from a register and provides it to the I/O structure in BCD format. Binary to BCD conversion can be used to drive BCD-encoded LED displays or preset external devices, such as high-speed counters.

The source of the binary data can be any valid register. Converted data is stored in any 16 I/O references whose first value is one greater than a multiple of 8 (for example, I0017, I0065, O0129). Because 16 references are required, the maximum point number of the second reference is either I0009 or O1009.

Every scan that power flows to the BIN/BCD function, the program converts the 16 digits of binary data beginning in the first reference into a BCD pattern. If the value in the register is greater than 9999, only the lower four digits are placed in the I/O and power flow is generated to indicate an error.

Entering a Binary to BCD Function

A Binary to BCD function can be placed in columns 1 to 8 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Shift/Move (F3) and then Binary/BCD Convert (F5).

```
  R****    *****
-| BIN  TO  BCD  |-
```

3. The cursor is at BIN. Using the numeric keypad, enter a register up to R1024. Then, press CTRL-E (or the Enter key).
4. The cursor moves to BCD. Enter a beginning address in the Main I/O Status table for the converted data. Then, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Convert BCD to Binary

The BCD to Binary function converts BCD data from the I/O structure into binary data and stores it in register memory. BCD/Binary conversion can be used to interface to BCD thumbwheels or external BCD electronics, such as high-speed counters or position encoders. Binary-Coded Decimal (BCD) values can display decimal (0-9) values one digit at a time. The table below shows standard coding for four wires, each of which can be ON or OFF (binary). The left column of the table shows the decimal digit represented by the possible states of the four wires.

	BCD Signal Definitions represented by ON Signal Values			
Digit	**8**	**4**	**2**	**1**
0	OFF	OFF	OFF	OFF
1	OFF	OFF	OFF	ON
2	OFF	OFF	ON	OFF
3	OFF	OFF	ON	ON
4	OFF	ON	OFF	OFF
5	OFF	ON	OFF	ON
6	OFF	ON	ON	OFF
7	OFF	ON	ON	ON
8	ON	OFF	OFF	OFF
9	ON	OFF	OFF	ON

Every scan that power flows to the BCD/BIN function, it converts the BCD values represented by the 16 I/O references (4 digits) starting at the first reference into a single binary number from 0000 - 9999. The binary number is then stored in the second reference.

The I/O reference used in the Convert BCD/Binary function (i.e., I0017) represents the value 1 (unity). The other I/O references increase in value as the reference increases (I0018 = 2, I0019 = 4, and so on). If the selected I/O references contain a BCD pattern that represents a value greater than 9, that value is used in the binary conversion, and power flows to indicate the error.

Entering a BCD to Binary Function

A BCD to Binary function can be placed in columns 1 to 8 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Shift/Move (F3) and then BCD/Binary Convert (F6).

```
  *****   R****
-| BCD  TO  BIN  |-
```

3. The cursor is at BCD. Using the numeric keypad, enter an address in the Main I/O status table as the reference for the BCD value. Then, press CTRL-E (or the Enter key).
4. The cursor moves to BIN. Enter a register up to R1024 as the destination for the binary data after the conversion. Then, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

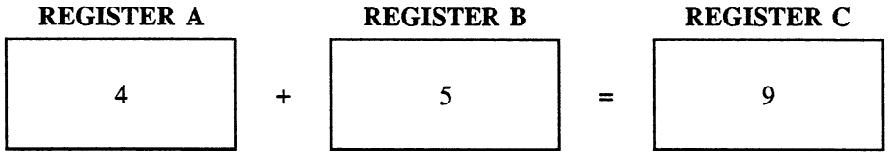
SECTION 4
Basic Arithmetic

This section is a reference to the Basic Arithmetic functions:

- Unsigned Addition (ADD)
- Unsigned Subtraction (SUB)
- Unsigned Compare

Unsigned Addition (Basic)

The Unsigned Addition function adds the value in one register to the value in another register and places the result in a third register.



Every scan that power flows to the ADD function, the program adds the content of register A to the content of register B and places the result in register C. Only the content of register C is altered by this function. If the sum of register A plus register B is outside the range 0 to 65,535, the function supplies power flow and the sum minus 65,536 is placed in register C.

Entering an Unsigned Addition Function

An Unsigned Addition function can be placed in columns 1 to 7 of the first line of a rung.

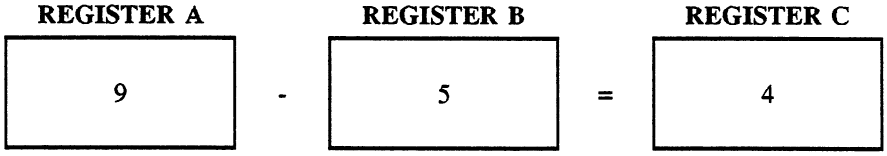
1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Basic Arithmetic (F4) and then ADD A+B=C (F1).

```
   R****    R****    R****
-|  A   +   B   =    C   |-
```

3. The cursor is at A. Using the numeric keypad, enter a register up to R1024 as the location of the first value to be added. Then, press CTRL-E (or the Enter key).
4. The cursor moves to B. Enter a register up to R1024 as the location of the second value. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Enter a register up to 1024 as the location where the result of the addition will be stored. Then, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Unsigned Subtraction

The Unsigned Subtraction function subtracts the value in one register from the value in another register and places the result in a third register.



Every scan that power flows to the SUBTRACT function, the program subtracts the content of register B from the content of register A and places the absolute value of the result in register C. Only the content of register C is altered by this function.

A zero or negative difference (meaning the content of register B is greater than the content of register A), causes power flow. Power flow is the only indicator of sign.

Entering an Unsigned Subtraction Function

An Unsigned Subtraction function can be placed in columns 1 to 7 of the first line of a rung.

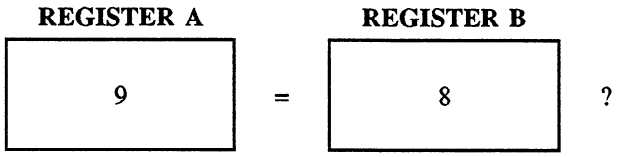
1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Basic Arithmetic (F4) and then SUB A-B=C (F2).

```
   R****    R****    R****
-| A    -   B    =   C   |-
```

3. The cursor is at A. Using the numeric keypad, enter a register up to R1024 as the location of the first value. Then, press CTRL-E (or the Enter key).
4. The cursor moves to B. Enter a register up to R1024 as the location of the value to be subtracted. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Enter a register up to R1024 as the location where the result of the subtraction will be stored. Then, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Unsigned Compare

The Unsigned Compare function compares the value in one register with the value in another.



Every scan that power flows to the COMPARE function, the program compares the content of register A to the content of register B. Power flow is output only if the two values are equal.

Entering an Unsigned Compare Function

An Unsigned Compare function can be placed in columns 1 to 7 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Basic Arithmetic (F4) and then Compare A:B (F4).

```
  R****   R****
-|  A   :   B  |-
```

3. The cursor is at A. Using the numeric keypad, enter a register up to R1024 as the location of the first value. Then, press CTRL-E (or the Enter key).
4. The cursor moves to B. Enter a register up to R1024 as the location of the value to be compared to the first value. Then, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

NOTE

For registers above R1024, use the Expanded Compare function, described in chapter 14, *Expanded Functions* (page 14-21).

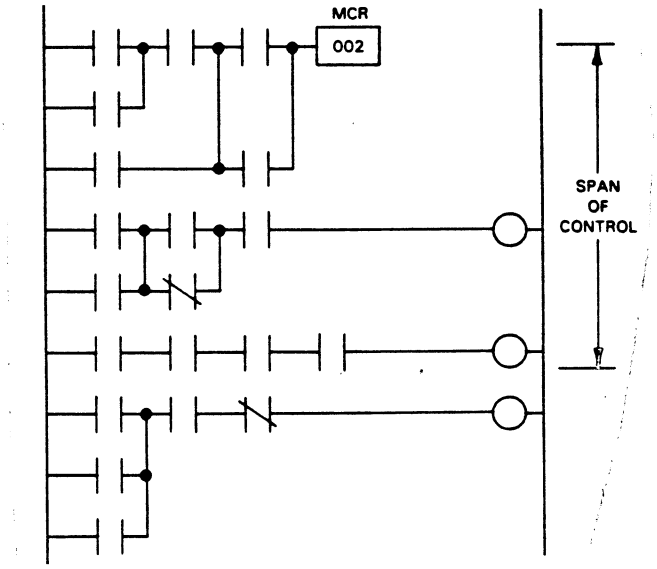
SECTION 5
Special Functions

This section is a reference to these Special functions:

- Master Control Relay (MCR)
- Skip
- No Operation
- End of Sweep
- Serial Communication Request
- Data Processing Unit Request

Master Control Relay

Every scan that power is received, a Master Control Relay (MCR) function causes a portion of the program after the MCR to be bypassed. Program execution resumes after the specified number of rungs, or at the next Return or End of Sweep command.



pcs6840271

When the MCR is active, all of the outputs associated with relay coils are turned off, unless overridden. Timers, counters, and latch coils remain in their last known state.

When power flow to the MCR is interrupted, the normal logic functions start with coils off and register values not altered.

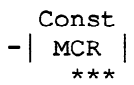
Only coils with valid output references are included in determining the area to be bypassed. Rungs that end in coils without references are also bypassed, but do not count toward the total number of rungs to be bypassed. If the number 0 is entered for the MCR, all logic to the next Return or End of Sweep is controlled by the MCR.

Master Control Relays and Skips (see below) can be used in building shift registers, stepping switches, and other logic functions.

Entering an MCR

An MCR can be placed in columns 1 through 9 of any line of a rung.

1. Enter the logic for the rung, leading to the MCR. Logic located before the MCR in the rung will execute normally, and will not be affected by the MCR.
2. Select Special Functions (F5) and then MCR (F1).



3. Using the numeric keypad, enter the number of referenced coils to be bypassed during the MCR. It must be a constant from 0 to 255. Enter 0 to have the MCR active to the next Return or End of Sweep function. After entering the number, press CTRL-E (or the Enter key).
4. When the rung is complete, press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Skip

A Skip function is similar to an MCR, except that coils are not turned off. They remain in their pre-skip condition until power flow is removed from the skip. Program execution resumes after the specified number of rungs, or at the next Return or End of Sweep command.

When the skip is active, all of the outputs associated with the timers, counters, one-shots, latches, or relays within its scope are frozen.

When power flow to the skip is interrupted, the normal logic functions start again. Logic is not reset.

Only coils with valid output references are included in determining the area to be bypassed. Rungs that end in coils without references are also bypassed, but do not count toward the total number of rungs to be bypassed. If the number 0 is entered for the skip, all logic to the next Return or End of Sweep is controlled by the skip.

MCRs and skips can be used in building shift registers, stepping switches, and other logic functions.

Entering a Skip

A skip can be placed in columns 1 through 9 of any line of a rung.

1. Enter the logic for the rung, leading to the skip. Logic located before the skip in the rung will execute normally and will not be affected by it.
2. Select Special Functions (F5) and then Skip (F2).

3. Using the numeric keypad, enter the number of referenced coils to be bypassed during the skip. It must be a constant from 0 to 255. Enter 0 to have the skip active to the next Return or End of Sweep function. After entering the number, press CTRL-E (or the Enter key).
4. When the rung is complete, press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

No Operation

The No Operation (No Op) function occupies memory that will later be used for a real function. A No Op can also be used to form a break between areas of logic. To enter a No Op into a program, select Special Functions (F5), then No Operation (F3).

```
-|NO OP|-
```

End of Sweep

The system places an End of Sweep function at the end of the main program. Two End of Sweeps indicate the end of the entire ladder logic.

Serial Communications Request

The SCREQ function can be used to initiate serial communication. Transfer may be to or from the CPU or a device (such as another CPU or the Logicmaster system) connected to port J1 or J2. This function will not operate if the SCREQ function has been disabled using the Status function.

Using the SCREQ function requires a knowledge of serial communications protocols and timing, and other information provided in the *Series Six Data Communications User's Manual* (GEK-25364). Refer to that manual when using the SCREQ function.

A serial communications request can be programmed with a register reference only. The parameters required to define the data transfer occupy seven consecutive registers.

The first scan that power flows to the function, the transfer occurs. Power flow is generated by the function if a previous request has been activated and is not completed (device busy). This request is queued and acted upon when the device is available, as long as power flow remains applied to the function.

Entering a Serial Communications Request Function

A SCREQ function can be placed in columns 1 to 9 of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Special Functions (F5) and then Serial Communications Request (F5).

```
  R****
-|SCREQ|-
```

3. The cursor is at SCREQ. Using the numeric keypad, enter a register up to R1018 as the first of the seven registers that will contain the data transfer parameters. Then, press CTRL-E (or the Enter key).
4. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Data Processing Unit Request

The DPREQ function can be used to cause a data transfer between the CPU and another module, such as a Loop Management Module, ASCII/BASIC Module, IOCCM Module, Bus Controller, or I/O Link Local Module.

The DPREQ function is used only with Normal I/O addressing. It cannot be used to specify Expanded I/O channel numbers. For an Expanded system, use the Window function, as described in chapter 10, *Expanded Functions Menu.*

A data processing unit request must be programmed with a register reference. This reference is the first of up to seven consecutive registers, called a command block. The command block contains the data transfer parameters.

DPREQ Command Block

Device Address + 1000	Register n
	Register n+1
	Register n+2
Command Parameters	Register n+3
	Register n+4
	Register n+5
	Register n+6

The content of the command block registers depends on the type of module using the DPREQ. Refer to the instructions provided with the module in order to program the DPREQ and to place data in the command block.

When power flows to a DPREQ, data transfer occurs between the device and the CPU. Power flow is generated by the function if a previous request has been activated and is not completed (device busy), or if an error occurs, such as a window timeout or an invalid operand. The current request is then queued and acted upon when the device is available, as long as power flow remains applied to the function. The DPREQ will not operate if disabled using the Status function.

Entering a Data Processing Request Function

A DPREQ function can be placed in columns 1 to 9 of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Special Functions (F5) and then Data Processing Request (F6).

```
  R****
-|DPREQ|-
```

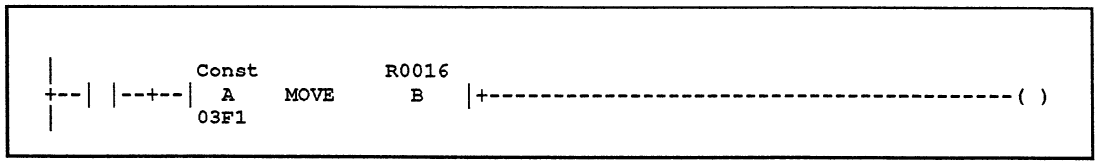
3. The cursor is at DPREQ. Using the numeric keypad, enter a register up to R1018 as the first of the seven registers (command block) that will contain the data transfer parameters. It must be a valid register. This first register will contain the address of the communicating device, plus 1000 decimal. You can enter this value now, or place it in the register using program logic. Using program logic, such as Data Move function, to place the value in the register, allows you to use a single DPREQ to communicate with more than one device.

   To enter the device address using program logic, complete the DPREQ now by pressing CTRL-E (or the Enter key). If, instead, you want to enter the device address now, move the work area cursor to the bottom line and enter the correct address value, referring to the table on the next two pages. After entering the value for the reference, press the Shift and Enter keys. That will place both the reference and the device address into the logic.
4. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

DPREQ Command Block

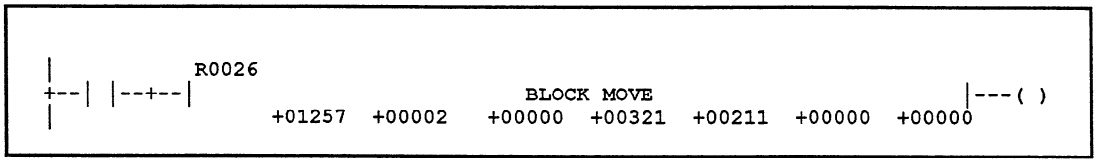
The first DPREQ register contains a value equal to the I/O address of the communicating device plus 1000 decimal, as explained above. The content of the other registers in the command block depends on the type of module communicating with the CPU. In some cases, not all the registers in the command block are used by the DPREQ. They can be used for other purposes in the program.

Communications parameters can be placed in the command block registers directly, or through program logic. For example, the device address value can be placed in the first register using a Data Move function prior to execution of the DPREQ:



In this example, the hexadecimal value 03F1 (corresponding to a module I/O address of 9-16) is moved into register R0016, which is the first register in a command block.

Similarly, values can be placed in all registers of the command block using a Block Move function that executes prior to the DPREQ.



Register Contents for the DPREQ Instruction

Refer to the table on the next two pages to enter the DPREQ device address. This value must be equal to the card address of the module plus 1000 (decimal).

DPREQ Device Address: The value you enter, either in decimal or hexadecimal. Each value is equal to the I/O address plus 1000 decimal.

I/O Point: The center column shows I/O addresses used by the module.

Dip Switch Position: The right column shows dip switch settings which correspond to the first I/O point listed in column 2.

DPREQ Device Address		Dip Switch Position						
Decimal (Hex)	I/O Point	7	6	5	4	3	2	1
1001 (03E9)	1-8							
1009 (03F1)	9-16							x
1017 (03F9)	17-24						x	
1025 (0401)	25-32						x	x
1033 (0409)	33-40					x		
1041 (0411)	41-48					x	x	
1049 (0419)	49-56					x	x	
1057 (0421)	57-64					x	x	x
1065 (0429)	65-72				x			
1073 (0431)	73-80				x		x	
1081 (0439)	81-88				x	x		
1089 (0441)	89-96				x	x	x	
1097 (0449)	97-104				x	x		
1105 (0451)	105-112				x	x	x	
1113 (0459)	113-120				x	x	x	
1121 (0461)	121-128				x	x	x	x
1129 (0469)	129-136			x				
1137 (0471)	137-144			x			x	
1145 (0479)	145-152			x		x		
1153 (0481)	153-160			x		x	x	
1161 (0489)	161-168			x	x			
1169 (0491)	169-176			x	x	x		
1177 (0499)	177-184			x	x	x		
1185 (04A1)	185-192			x	x	x	x	
1193 (04A9)	193-200			x	x			
1201 (04B1)	201-208			x	x		x	
1209 (04B9)	209-216			x	x	x		
1217 (04C1)	217-224			x	x	x	x	
1225 (04C9)	225-232			x	x	x		
1233 (04D1)	233-240			x	x	x	x	
1241 (04D9)	241-248			x	x	x	x	
1249 (04E1)	249-256			x	x	x	x	x
1257 (04E9)	257-264		x					

DPREQ Device Address		Dip Switch Position						
Decimal (Hex)	I/O Point	7	6	5	4	3	2	1
1265 (04F1)	265-272		x				x	
1273 (04F9)	273-280		x			x		
1281 (0501)	281-288		x			x	x	
1289 (0509)	289-296		x		x			
1297 (0511)	297-304		x		x	x		
1305 (0519)	305-312		x		x	x		
1313 (0521)	313-320		x		x	x	x	
1321 (0529)	321-328		x	x				
1329 (0531)	329-336		x	x		x		
1337 (0539)	337-344		x	x	x			
1345 (0541)	345-352		x	x	x	x		
1353 (0549)	353-360		x	x	x			
1361 (0551)	361-368		x	x	x	x		
1369 (0559)	369-376		x	x	x	x		
1377 (0561)	377-384		x	x	x	x	x	
1385 (0569)	385-392		x	x				
1393 (0571)	393-400		x	x			x	
1401 (0579)	401-408		x	x		x		
1409 (0581)	409-416		x	x		x	x	
1417 (0589)	417-424		x	x	x			
1425 (0591)	425-432		x	x	x	x		
1433 (0599)	433-440		x	x	x	x		
1441 (05A1)	441-448		x	x	x	x	x	
1449 (05A9)	449-456		x	x	x			
1457 (05B1)	457-464		x	x	x		x	
1465 (05B9)	465-472		x	x	x	x		
1473 (05C1)	473-480		x	x	x	x	x	
1481 (05C2)	481-488		x	x	x	x		
1489 (05D1)	489-496		x	x	x	x	x	
1497 (05D9)	497-504		x	x	x	x	x	
1505 (05E1)	505-512		x	x	x	x	x	x
1513 (05E9)	513-520	x						
1521 (05F1)	521-528	x					x	

DPREQ Device Address		Dip Switch Position						
Decimal (Hex)	I/O Point	7	6	5	4	3	2	1
1529 (05F9)	529-536	x				x		
1537 (0601)	537-544	x				x	x	
1545 (0609)	545-552	x			x			
1553 (0611)	553-560	x			x		x	
1561 (0619)	561-568	x			x	x		
1569 (0621)	569-576	x			x	x	x	
1577 (0629)	577-584	x		x				
1585 (0631)	585-592	x		x			x	
1593 (0639)	593-600	x		x		x		
1601 (0641)	601-608	x		x		x	x	
1609 (0649)	609-616	x		x	x			
1617 (0651)	617-624	x		x	x		x	
1625 (0659)	625-632	x		x	x	x		
1633 (0661)	633-640	x		x	x	x	x	
1641 (0669)	641-648	x	x					
1649 (0671)	649-656	x	x				x	
1657 (0679)	657-664	x	x			x		
1665 (0681)	665-672	x	x			x	x	
1673 (0689)	673-680	x	x		x			
1681 (0691)	681-688	x	x		x		x	
1689 (0699)	689-696	x	x		x	x		
1697 (06A1)	697-704	x	x		x	x	x	
1705 (06A9)	705-712	x	x	x				
1713 (06B1)	713-720	x	x	x			x	
1721 (06B9)	721-728	x	x	x		x		
1729 (06C1)	729-736	x	x	x		x	x	
1737 (06C9)	737-744	x	x	x	x			
1745 (06D1)	745-752	x	x	x	x		x	
1753 (06D9)	753-760	x	x	x	x	x		
1761 (06E1)	761-768	x	x	x	x	x	x	

DPREQ Device Address		Dip Switch Position						
Decimal (Hex)	I/O Point	7	6	5	4	3	2	1
1769 (06E9)	769-776	x	x					
1777 (06F1)	777-784	x	x					x
1785 (06F9)	785-792	x	x				x	
1793 (0701)	793-800	x	x				x	x
1801 (0709)	801-808	x	x			x		
1809 (0711)	809-816	x	x			x		x
1817 (0719)	817-824	x	x			x	x	
1825 (0721)	825-832	x	x			x	x	x
1833 (0729)	833-840	x	x		x			
1841 (0731)	841-848	x	x		x			x
1849 (0739)	849-856	x	x		x		x	
1857 (0741)	857-864	x	x		x		x	x
1865 (0749)	865-872	x	x		x	x		
1873 (0751)	873-880	x	x		x	x		x
1881 (0759)	881-888	x	x		x	x	x	
1889 (0761)	889-896	x	x		x	x	x	x
1897 (0769)	897-904	x	x	x				
1905 (0771)	905-912	x	x	x				x
1913 (0779)	913-920	x	x	x			x	
1921 (0781)	921-928	x	x	x			x	x
1929 (0789)	929-936	x	x	x		x		
1937 (0791)	937-944	x	x	x		x		x
1945 (0799)	945-952	x	x	x		x	x	
1953 (07A1)	953-960	x	x	x		x	x	x
1961 (07A9)	961-968	x	x	x	x			
1969 (07B1)	969-976	x	x	x	x			x
1977 (07B9)	977-984	x	x	x	x		x	
1985 (07C1)	985-992	x	x	x	x		x	x
1993 (07C9)	993-1000	x	x	x	x	x		

x - means the switch is in the OPEN position (depressed to the left).

Using a DPREQ to Communicate with the Genius I/O System

The DPREQ function can be used to communicate with a Genius I/O system with Normal (Non-Expanded) addressing. For a system with Expanded addressing, use the Window function instead, as described in chapter 14, *Expanded Functions*.

For communication with a bus controller, the DPREQ command block registers have the following parameters:

DPREQ Command Block for Genius I/O

B/C Address + 1000	Register n
Command Number	Register n+1
DPREQ Status Number	Register n+2
B/C or Block I/O Address	Register n+3
Register to Store Data	Register n+4
Length of Data in Bytes	Register n+5
Not used	Register n+6

Parameters of the DPREQ Command Block for a Bus Controller

For communication with the Genius I/O system, the content of the DPREQ command block may be:

Register 1: The backplane address of the resident bus controller. This is equal to the bus controller's first status reference (+ 1000 for a DPREQ).

Register 2: A number of commands can be performed from the CPU. The content of the second register indicates the command number of the command to be performed:

Command Number	Description
1	Idle (do nothing).
2	Read configuration of I/O Block or resident bus controller.
3	Write configuration of I/O Block or resident bus controller. (Write Configuration of bus controller is also used to send global data.)
4	Read diagnostic data of I/O Block or bus controller (into CPU registers).
5	Reserved.
6	Reserved.
7	Read analog status (all analog inputs from a block into the CPU registers).
8	Reserved.
9	Read status table reference.
10	Reserved.
11	Switch BSM In a dual bus system, this command will switch a BSM to a specified bus position.
12	Send Datagram.
13	Receive Datagram.

Register 3: Status Code. This register should first be cleared to zero by the CPU. It will be loaded by the bus controller at the end of the DPREQ instruction (when status is available). Its content may be:

Command Number	Description
0	Not accepted, CPU or bus controller busy with previous DPREQ.
1	Command in process but not completed.
2	Command completed successfully.
12	Command terminated due to syntax error.
20	Command terminated due to data transfer error.

The bus controller verifies the command block for valid command syntax and the absence of a command of that type already in execution. If any syntax errors exist, the bus controller writes the error code (12) into the status code in the third register of the command block in the CPU.

Register 4: The fourth register containing the I/O block status table address or bus controller status table address indicates the starting reference of the specific device (which may be an I/O block or the resident bus controller) involved in the command for command numbers 2-7. (Bits 0-9 represent the I/O address, plus one bit (bit 15) if the command is to an outputs-only I/O block). For command numbers above 8, the fourth register indicates the device number (serial bus address) of the I/O block or bus controller.

Register 5: The fifth register points to an address in CPU memory. For commands 2, 4, 7, and 9, this address is where data received following Read commands will be placed in the CPU. For commands 3 and 12, this register contains the address where the data to be sent to the Genius I/O subsystem begins in the CPU (header and data for command 12). For command 13, this register contains the address where the information to be read (header) resides. The actual data will be received at the address pointed to by register 8.

Register 6: This register is used for commands 12 and 13 (Datagrams) only. It contains the length in bytes of the message to be transmitted.

Register 7: Used for commands 12 and 13 only, to specify the command code of the datagram message being executed. It may be Read Device, Write Device, Write Point, or Assign Monitor. If executing command 13, the remaining part of the command code resides in register 10.

Register 8: Used for command 13 only, to specify command code for the first register in the CPU where the message (acknowledge header plus data) which is received in reply to the datagram will be stored.

Register 9: Used for command 13 only, to specify the message received length. This value is returned from the bus controller when the DPREQ command is complete.

Register 10: Used for command 13 only, to specify the remaining part of the command code in register 7.

To use Genius I/O commands, refer to the *Genius I/O System User's Manual* (GEK-90486) for more information.

SECTION 6
Data Move Functions

Data Move functions are used to transfer data in memory. The specified data is copied from one location to another. It exists unchanged in the original location, and writes over any data already stored in the specified location.

Use this section as a reference to the Data Move functions:

- A to B Move
- Right 8 Move
- Left 8 Move
- Block Move

A to B Move

The A to B Move function copies the content of one 16-bit storage location to another. Every scan that power flows to the A to B Move function, the data is copied from reference A to reference B. The data in A is retained, and the previous content of B is lost. Power flow is generated whenever the function is active.

Entering an A to B Move Function

An A to B Move function can be placed in columns 1 through 8 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Functions (F7), Data Move (F1), and then A to B Move (F1).

```
 *******          *******
-|   A     MOVE      B  |-
```

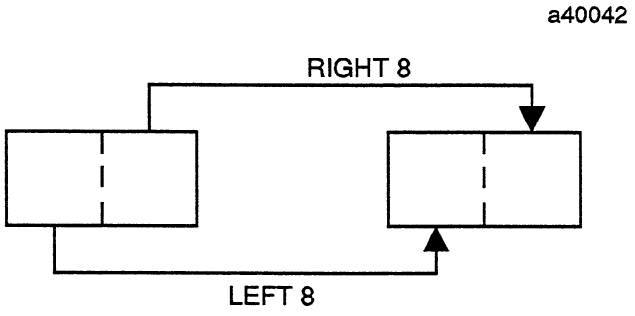
3. The cursor is at A. Using the numeric keypad, type in the source of the data. In the Advanced function set, it may be any available register (R0001 -R1024), Main or Auxiliary I/O reference beginning on a byte boundary, or a constant to +16,383. In addition, in the Expanded or Expanded II function set, it may also be any Expanded I/O reference. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Enter the destination of the copied data. In the Advanced function set, it may be any available register (R0001 - R1024), or Main or Auxiliary I/O reference beginning on a byte boundary. In addition, in the Expanded or Expanded II function set, it may also be any Expanded I/O reference. After entering the reference, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Move Right or Left 8 Bits

Either of the following two functions can be used to copy half of the content of a 16-bit storage location. This might be used to force pointers, and to cause data to be moved non-sequentially.

The Move Right 8 Bits function copies the right 8 bits of one 16-bit storage location to the right 8 bits of another.

The Move Left 8 Bits function copies the left 8 bits of one 16-bit storage location to the left 8 bits of another.



Every scan that power flows to a Move 8 Bits function, the data is copied from reference A to reference B. The data in reference A is retained, and the previous content of the corresponding 8 bits of reference B is lost. Power flow is generated whenever the function is active.

Be careful when using Move Left 8 to copy the 8 high-order bits of a constant value. Fixed values from 0000 to 3FFF hex and from 8000 to FFFF hex will operate as expected. Values from 4000 to 7FFF hex cannot be entered.

Entering an 8 Bits Move Function

An 8 Bits Move function can be placed in columns 1 through 8 of the first line of a rung.

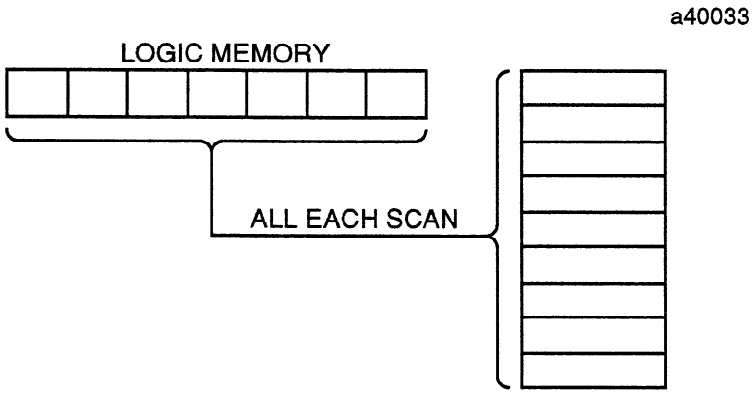
1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Functions (F7), Data Move (F1), and then either Move Left 8 (F2) or Move Right 8 (F3).

```
 *******          *******
-|  MOVE LEFT   8 BITS  |-
```

3. The cursor is at MOVE. Using the numeric keypad, enter any available register or I/O reference as the source of the data. Then, press CTRL-E (or the Enter key).
4. The cursor moves to BITS. Enter any available register or I/O reference as the destination of the copied data. After entering the reference, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Block Move

Block Move can be used to load up to 7 constant values directly from the program into a specified destination, every scan that power flows to this function. The Block Move function generates power flow whenever it is active.



Entering a Block Move

The Block Move function can only be placed in columns 1 or 2 of a rung. Enter one contact if required to control execution of the Block Move function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.

1. Select Advanced Mnemonic Group (F7), Data Move (F1), and then Block Move (F4).

```
 *******
-|                                  BLOCK MOVE                          |-
          +00000  +00000  +00000  +00000  +00000  +00000  +00000
```

2. Using the numeric keypad, enter any available register or I/O reference as the destination of the data from the program. This destination must be the first of seven consecutive storage locations. After entering the reference, press CTRL-E (or the Enter key).
3. Move the cursor above the first constant value.
4. Press CTRL-S (or the Select key) to move the work area banner to the bottom (value) line. Using the numeric keypad, type in the first constant value. Constant values used with this function must be between +32,767 and -32,768.
5. After typing in the value, press CTRL-E (or the Enter key) to place it in the rung.
6. Move the cursor to the next value location. Type the next value into the work area. Press CTRL-E (or the Enter key) to place it in the rung.
7. Enter the rest of the values.
8. Press the Coil (F7) function key to enter a coil at the end of the line.
9. When the line of logic is complete, press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

SECTION 7
Signed Arithmetic Functions

This section is a reference to the Signed Arithmetic functions:

- Signed Addition
- Signed Subtraction
- Double Precision Addition
- Double Precision Subtraction
- Signed Multiplication
- Signed Division
- Greater Than

Floating Point Math functions are explained in the next chapter.

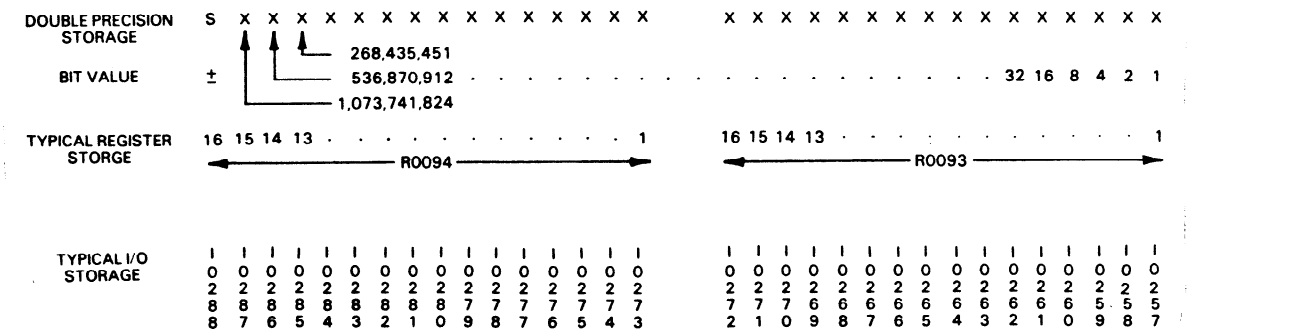
How Signed Arithmetic Functions are Stored

All advanced mathematical functions handle signed values, which can be displayed as either positive or negative values. Signed values are stored in 16 contiguous bits of memory, with the leftmost bit used as the sign bit. Single register signed values can range from -32,768 to +32,767. Constants are allowed by most advanced math functions, but not in all positions.

Double precision values may be used to temporarily store 32 bits of memory storage. Double precision values are always signed. The valid range for double precision values is -2,147,483,648 to +2,147,483,647.

The illustration below shows how double precision values may be stored. The lower part of the illustration shows how the double precision value could be stored in 32 consecutive I/O references. The I/O references shown are an example only.

pcs6840280



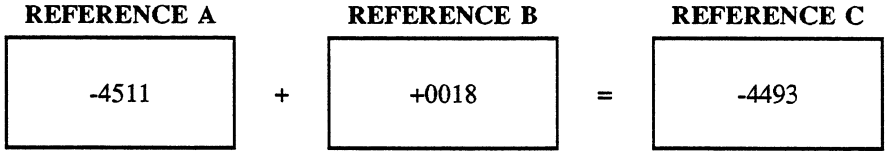
Using Values from Basic Math Functions

Positive values from the basic math functions can be used as single precision positive values. No further conversion is required if the value is not more than 32,767. If the value is greater than 32,767, it must be stored as a double precision value before it can be used as a signed number.

To use a negative value from the basic math functions with the one of the advanced math functions, first subtract the number from zero using the Signed Subtraction (SUBX) function.

Signed Addition (ADDX)

The Signed Addition function adds the value in one reference to the value in another reference and places the result in a third reference.



Every scan that power flows to the ADDX function, the program adds the content of reference A to the content of reference B and places the result in reference C. Only the content of reference C is altered by this function. If the sum is greater than -32,768 or +32,767, the function supplies power flow and the value -32,768 or +32,767 is placed in reference C.

Entering a Signed Addition Function

A Signed Addition function can be placed in columns 1 to 7 of the first line of a rung.

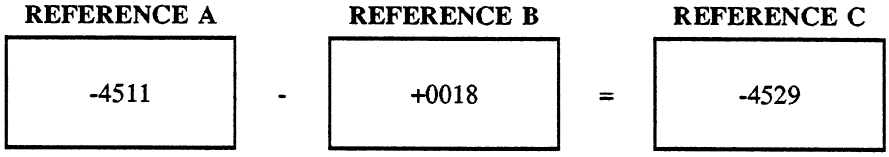
1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), and then Signed Addition (F1).

```
 ******* ******* *******
-|   A ADDX  B  =    C |-
```

3. The cursor is at A. Using the numeric keypad, enter any available register or I/O reference as the location of the first value to be added, or a constant to ±16,383. Then, press CTRL-E (or the Enter key).
4. The cursor moves to B. Enter any available register or I/O reference as the location of the second value to be added, or a constant to ±16,383. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Enter any available register or I/O reference as the location where the result of the addition will be stored. After entering the reference, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Signed Subtraction (SUBX)

The Signed Subtraction function subtracts the value in one reference from the value in another reference and places the result in a third reference.



Every scan that power flows to the SUBX function, the program subtracts the content of reference B from the content of reference A and places the result in reference C. Only the content of reference C is altered by this function. The function supplies power flow only if the value is 0 or less.

Entering a Signed Subtraction Function

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), and then Signed Subtraction (F2).

```
 ******* ******* *******
-|   A SUBX  B  =    C |-
```

3. The cursor is at A. Using the numeric keypad, enter any available register or I/O reference as the location of the first value, or a constant to ±16,383. Then, press CTRL-E (or the Enter key).
4. The cursor moves to B. Enter the reference for the value to be subtracted from the value in reference A. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Enter any valid register or I/O reference as the location where the result of the subtraction will be stored. Then, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Double Precision Addition (DPADD)

The Double Precision Addition function adds the value in one reference to the value in another reference and places the result in a third reference.

Every scan that power flows to the DPADD function, the program calculates the result using double precision mathematics. Automatic carries are performed between the two registers or groups of I/O, if necessary. The result is placed in reference C. Only the content of reference C is altered by this function.

When using double precision values, the number must be broken into two parts, the high order (representing values above 65,537) and the low order (representing values below 65,536). The digit 1 has the value 1 in low order storage, and the value 65,536 in high order storage. In the Series Six PLC CPU, the reference used in the double precision function represents the low order storage, and the higher references are the high order storage.

The function outputs power flow if the result is greater than the capacity of the double precision storage. Then, +2,147,483,647 or -2,147,483,648 (depending on whether the overflow is in the positive or negative direction) is placed in the storage location.

Entering a Double Precision Addition Function

A Double Precision Addition function can be placed in columns 1 to 4 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), Double Precision Arithmetic (F5), and then Signed Double Precision Addition (F1).

3. The cursor is at A. Using the numeric keypad, type in reference of the first value to be added. It is a double precision value, and may be any available register or I/O reference, or a constant from +1,073,741,823 to -1,073,741,824. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same paremeters, enter the reference for the second value to be added. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Reference C is a double precision reference. It may be any available register or I/O reference. Press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Double Precision Subtraction (DPSUB)

The Double Precision Subtraction function subtracts the value in one reference from the value in another reference and places the result in a third reference.

Every scan that power flows to the DPSUB function, the program subtracts the value in reference A from the value in reference B using double precision mathematics. The result is placed in reference C. Only the content of reference C is altered by this function. For more information on the way double precision values are stored, refer to the preceding explanation of the Double Precision Addition function.

The Double Precision Subtraction function outputs power flow if the result is greater than the capacity of the double precision storage. Then, the value +2,147,483,647 or -2,147,483,648 (depending on whether the overflow is in the positive or negative direction) is placed in the storage location.

Entering a Double Precision Subtraction Function

A Double Precision Subtraction function can be placed in columns 1 to 4 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), Double Precision Arithmetic (F5), and then Signed Double Precision Subtraction (F2).

```
 *******          *******          *******
-|   A    DPSUB      B      =        C          |-
```

3. The cursor is at A. Using the numeric keypad, type in reference of the first value. It is a double precision value, and may be any available register or I/O reference, or a constant from +1,073,741,823 to -1,073,741,824. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the reference for the value to be subtracted from reference A. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Reference C is a double precision reference. It may be any available register or I/O reference. After entering the reference, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Signed Multiplication (MPY)

The Signed Multiplication function multiplies the value in one reference by the value in another reference and places the result in a third reference.

Every scan that power flows to the function, the system multiplies the value in reference A by the value in reference B and places the signed result in reference C. The possible sign of the result is shown below.

Reference A	Reference B	Reference C
+	+	+
+	-	-
-	+	-
-	-	+

Since there are no error conditions to be verified, the Signed Multiplication function always outputs power flow when active.

Entering a Signed Multiplication Function

A Signed Multiplication function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), and then Signed Multiplication (F3).

```
 ******* ******* *******
-|   A MPY   B  =    C       |-
```

3. The cursor is at A. Using the numeric keypad, type in reference of the first value. It is a single precision signed value, and may be any available register or I/O reference, or a constant to +32,767. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the reference for the value to be multiplied by reference A. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Reference C is a double precision reference. It may be any available register or I/O reference. After entering the reference, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Signed Division (DVD)

The Signed Division function divides the value in one reference by the value in another reference and places the result in a third and fourth reference. Every scan that power flows to the function, the content of reference A is divided by the content of reference B. The quotient is placed in QUO and the whole number remainder is placed in REM. The possible signs are shown below.

Reference A	Reference B	Quotient	Remainder
+	+	+	+
+	-	-	+
-	+	-	-
-	-	+	-

If the quotient is too large to fit in one register (including division by zero), the DVD function outputs power flow, and no storage values are altered.

Entering a Signed Division Function

A Signed Division function can be placed in columns 1 to 5 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), then Expanded Arithmetic (F2), then Signed Division (F4). The Signed Division display appears.

```
 *******           ******* ******* *******
-|   A   DVD          B      QUO     REM |-
```

3. The cursor is at A. Using the numeric keypad, type in reference of the first value. It is any available register or I/O reference, or a constant to +1,073,741,823. After entering any reference press CTRL-E (or the Enter key).
4. The cursor moves to B. Enter the reference for the single precision signed value to be divided by reference A. It may be any available register or I/O reference, or a constant to +16,383. Press CTRL-E (or the Enter key).
5. The cursor moves to QUO. Enter a reference to receive the single precision, signed quotient of the division. It may be any available register or I/O reference. Press CTRL-E (or the Enter key).
6. The cursor moves to REM. Enter a reference to receive the single precision, signed remainder of the division. The reference may be any available register or I/O reference. After entering the reference, press CTRL-E (or the Enter key).
7. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Greater Than

The Greater Than function compares two double precision signed values.

Every scan that power flows to the Greater Than function, the program compares the content of reference A to the content of reference B. Comparison is based on the double precision signed values of the contents.

Power flows only if the first value (A) is larger than the second.

Some example comparisons are shown below.

Reference A	Reference B	Power Flow ?
+57,001	+57,000	yes
+57,000	+57,001	no
+225	+225	no
-500	-600	yes
-600	-500	no

Entering a Greater Than Function

A Greater Than function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), Double Precision Arithmetic (F5), and then Greater Than (F3).

```
 *******          *******
-|    A GREATER THAN  B        |-
```

3. The cursor is at A. Using the numeric keypad, type in the reference of the first value, which is a double precision signed value. The reference may be any available register or I/O reference, or a constant to ±1,073,741,823. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the reference of the second value, which is a double precision signed value. Then, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

SECTION 8
Table Move Functions

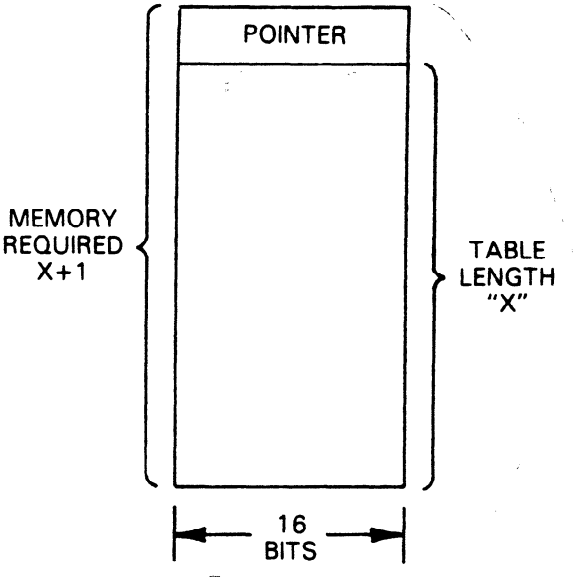
Table Move functions transfer data from one memory location to another. The data remains unchanged in the original location, and writes over any data already stored in the specified location.

This section is a reference to the Table Move functions.

- Source to Table Move
- Table to Destination Move
- Table to Table Move
- Extended Table Move

A table is a group of 16-bit data storage locations, either registers or consecutive I/O addresses. The data in a table is retained upon power failure. A table begins with a pointer, which controls the transfer of data in or out of the table. In addition to the pointer, the table may contain from 1 to 255 data locations.

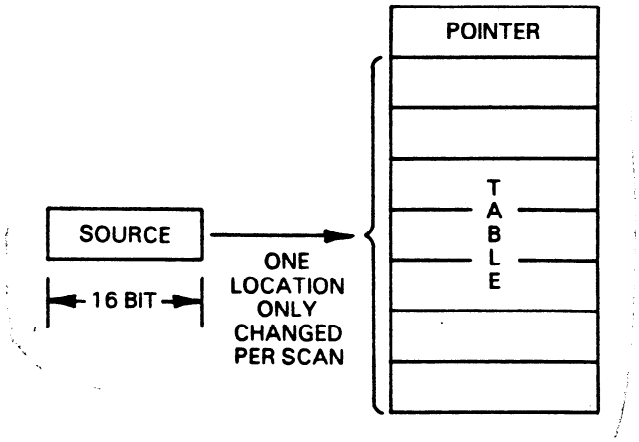
pcs6840281



Source to Table Move

The Source to Table Move copies successive 16-bit values from a single storage location into a table.

pcs6840282



Every scan that power flows to this function, the pointer increments by one, and the contents of the source memory storage location are transferred to the next location in the table. If the pointer value is greater than the specified table length or is zero, it is automatically reset to 1 before the data is transferred.

The data in the source location is copied into the location in the table which is currently indicated by the pointer. The value 1 indicates the first location, immediately after the pointer location. Therefore, the location in the table that receives the data is equal to the current pointer value plus the original pointer address. For example, if the address of the pointer is R0360, and the value contained in the pointer is 5, the data will be loaded into table location R0365.

If I/O addresses are used, multiply the value in the pointer by 16 to determine the location of the data. For example, if the address of the pointer is I0721 (within the 16 bits I0722-I0736), and the value contained in the pointer is 5, multiplying 5 x 16 = 90. The location of the data will be I0801 to I0816.

If the source value does not change, successive locations in the table will be loaded with the same value. To load values into a single location in the table, power flow should operate for only one scan. This may be done with a one-shot coil.

If the value of the pointer is exactly equal to the table length, the Source to Table Move function supplies power flow.

Entering a Source to Table Move

The Source to Table Move function can be placed in columns 1 to 7 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Table Move (F3), and then Source to Table Move (F1).

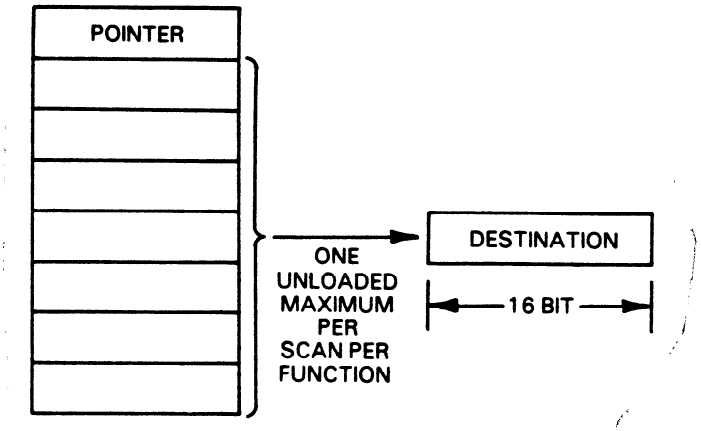
```
 ******* *******  Const
-| SRC-TO-TABLE      LEN |-
  +00000   00000    000
```

3. The cursor is at SRC. Using the numeric keypad, type in the source of the data. It may be any available register or I/O reference (beginning on a byte boundary). After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to TABLE. Using the same parameters, enter the address of the pointer. Then, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the table. It must be a constant from 1 to 255. The length must not go beyond the maximum available address. After entering the length, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Table to Destination Move

The Table to Destination Move copies successive 16-bit values from a table to a single storage location.

pcs6840283



Every scan that power flows to this function, the pointer increments by one. If the value of the pointer is exactly equal to the table length, the Source to Table Move function outputs power flow. The pointer then indicates the single location in the table to be copied to the destination.

The value 1 indicates the first location, immediately after the pointer location. Therefore, the location in the table that supplies the data is equal to the current pointer value plus the original pointer address. For example, if the address of the pointer is R0753, and the value contained in the pointer is 33 after incrementing, the data will be supplied by table location R0786.

When the bottom of the table is reached, the function automatically recycles to the top.

Unless controlled by a one-shot coil or similar logic, at every scan the next value in the table is copied to the destination. Therefore, the content of the destination will change with every scan.

Entering a Table to Destination Move

The Table to Destination Move function can be placed in columns 1 to 7 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Table Move (F3), and then Table to Destination Move (F2).

```
  ******* *******  Const
-| TABLE-TO-DEST     LEN |-
                     000
```

3. The cursor is at TABLE. Using the numeric keypad, type in the address of the pointer. It may be any available register or I/O reference (beginning on a byte boundary). After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to DEST. Using the same parameters, enter the destination for the data. Then, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the table. It may be a constant from 1 to 255, if the destination is a register, or 1 to 63 if the destination is the I/O tables. If necessary, the value for length can be calulated using this formula:

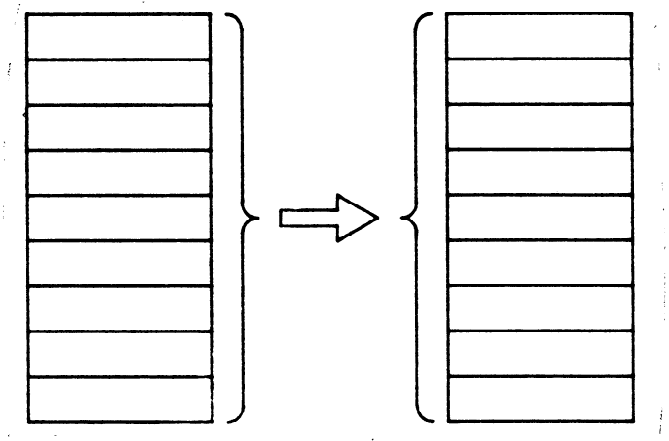
   Maximum length = 63 - ((I/O address - 1) divided by 16)

   After entering the length, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Table to Table Move

The Table to Table Move copies all the values in a table to another table.

a40032



Because this function copies an entire table at one time, no pointer is needed. Every scan the Table to Table Move function receives power flow, the entire table is copied to the destination table. This function can be used to call up recipes, standards, report data, or other information. Register data can be copied to either register or I/O locations, or vice versa. Unless controlled by a one-shot coil or similar logic, at every scan the entire table is copied.

Power flow is generated whenever power is received.

Entering a Table to Table Move

The Table to Table Move function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Table Move (F3), and then Table Move (F3).

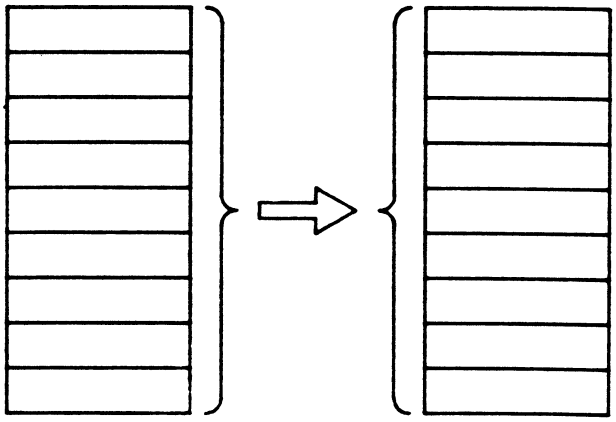
```
  *******           *******  Const
-| TBL A     MOVE    TBL B      LEN |-
                               000
```

3. The cursor is at TBL A. Using the numeric keypad, type in the beginning address of source table. It may be any available register or I/O reference (beginning on a byte boundary). After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to TBL B. Using the same parameters, enter the address of the destination for the table data. Then, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the table. It must be a constant from 1 to 255, within the maximum reference address. After entering the length, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Extended Table Move

The Extended Table Move is used like the Table to Table Move, to transfer data between one or more successive 16-bit storage locations. Without the Expanded Function set, it is the only Table Move function that can copy data in and out of registers above R1024. With the Expanded Function set, other functions can also address the higher registers.

a40032



The Extended Table Move can be used to store recipes, standards, report, or other large amounts of data. Register data can be copied to either register or I/O locations, or vice versa. The Table Move Extended function does not use a pointer. Every scan power is received, the entire table is copied to the destination table. Power flow is generated whenever power is received.

Entering an Extended Table Move

The Extended Table Move function can be placed in columns 1 to 5 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Table Move (F3), and then Extended Table Move (F4).

```
 *******          ******* *******
-|   A  MOVE TBL EXT B     LEN |-
```

3. The cursor is at A. Using the numeric keypad, type in the beginning address of the source table. It may be any available register or I/O reference beginning on a byte boundary. (To use indirect references for table A, first press the A/Bin key. Then enter the beginning address of the table where the addresses of the source data will be stored). After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the beginning address of the destination for the table data. (To use indirect references for table B, first press the A/Bin key. Then enter the beginning address of the table where the addresses for the destination data will be stored). After entering the reference, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the table. It may be a constant from 1 to 255, or any available register or I/O address (beginning on a byte boundary). After entering the length, press CTRL-E (or the Enter key).

6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Indirect Register Referencing

The Extended Table Move function can also be used with indirect references, if the CPU firmware is higher than version 104. Indirect referencing means that instead of containing actual data, one or both of the tables contains register addresses which "point" to the data, usually a register beyond 1024. The pointer should always be between 1 and the last available register.

1. Move the contents of R100 - R109 to R2000 - R2009.

```
 R100            IR0001 Const
| A   MOVE TBL EXT B    LEN   |-
                 2000   +010
```

2. Move the contents of R2000 - R2009 to R100 - R109.

```
 IR2000           R100 Const
| A   MOVE TBL EXT B    LEN   |-
  2000                 +010
```

3. Move the contents of R2000 - R2009 to R3000 - R3009.

```
 IR0001          IR0002 Const
| A   MOVE TBL EXT B    LEN   |-
  2000            3000   10
```

This powerful function allows the source and/or destination table to be dynamically programmed, to fit the requirements of the application.

SECTION 9
List Functions

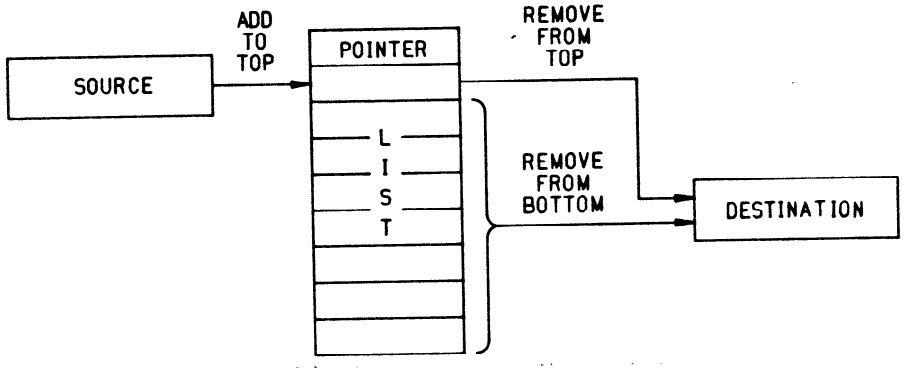
Use this section as a reference to the List functions:

- Add to Top
- Remove from Bottom
- Remove from Top
- Sort

List functions transfer data in memory. The specified data is copied from one location to another. It exists unchanged in the original location, and writes over any data already stored in the specified location.

A list is similar to a table. It consists of a sequence of adjacent 16-bit data storage locations. The list pointer, however, does not control the location of the stored data. Data may only be copied into the first location of a list, or copied out of the first or last location.

a40034



Add to Top

The Add to Top function copies a 16-bit value from a specified source to the first location in a list. This is the address immediately after the pointer.

Every scan that power is received by this function, the pointer is examined. If its value is less than the total length of the list, the value is copied into the first location. All other values move down one address. When the list is full, no further values are added to the list. The Add to Top function outputs power whenever the pointer equals the value of LEN.

Entering an Add to Top Function

The Add to Top function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), List Functions (F4), and then Add to Top (F1).

```
 *******          ******** Const
-| SRC ADD-TO-TOP  LIST     LEN |-
                            ***
```

3. The cursor is at SRC. Using the numeric keypad, type in the address of the data to be copied. It may be any available register or I/O reference (beginning on a byte boundary). After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to LIST. Enter the beginning address for the list, which will be the location of the pointer. This address may be a register up to 1 less than the Scratch Pad designated register size, or an I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the list. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Remove from Bottom

The Remove from Bottom function copies a 16-bit value from the last location on a list to a specified destination. When this function is used on a list filled with the Add to Top function, the result is a first in/first out queue. This function is frequently used to track items on a conveyor, where the first item on the conveyor is the first one to come off.

Every scan that power is received by this function, the pointer is examined. If its value is not zero, the data indicated by the pointer is copied to the destination address, and the pointer is decremented.

Note that this function does not actually move data in the list. That is done by the Add to Top function, which also increments the pointer. The Remove from Bottom function simply copies a value out and then decrements the pointer.

This function outputs power whenever the pointer is equal to zero.

Entering a Remove from Bottom Function

The Remove from Bottom function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), List Functions (F4), and then Remove from Bottom (F2).

```
 *******          ******** Const
-| LIST REM-FR-BOT DEST     LEN |-
                            ***
```

3. The cursor is at LIST. Using the numeric keypad, type in the beginning address for the list, which is the location of the pointer. This address may be a register up to 1 less than the Scratch Pad designated register size, or an I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to DEST. Enter the address of the destination where the data will be copied. It may be any available register or I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the list. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Remove from Top

The Remove from Top function copies a 16-bit value from the first location on a list to a specified destination. When this function is used on a list filled with the Add to Top function, the result is a last in/first out stack.

Every scan that power is received by this function, the pointer is examined. If its value is not zero, the data at the top of the list is copied to the destination address, all other entries are moved up one position, and the pointer is decremented.

This function outputs power whenever the pointer is equal to zero.

Entering a Remove from Bottom Function

The Remove from Bottom function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), List Functions (F4), and then Remove from Top (F3).

```
 *******         ******** Const
-| LIST REM-FR-TOP DEST     LEN |-
                            ***
```

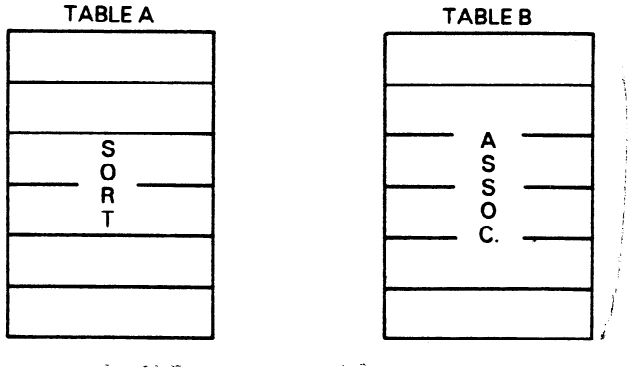
3. The cursor is at LIST. Using the numeric keypad, type in the beginning address for the list, which is the location of the pointer. This address may be a register up to 1 less than the Scratch Pad designated register size, or an I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to DEST. Enter the address of the destination where the data will be copied. It may be any available register, or an I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the list. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Sort

Program a Sort to reorder all the values contained in a list in ascending numerical order. After the data is sorted, it is written over the original data in the list addresses. You may want to first copy the data to be sorted into another location, so the original organization of the data is not lost. Unless controlled by a one-shot coil or similar logic, the sort occurs at every scan. Power flow is generated whenever power is received.

In addition to the list of values that are sorted, the Sort function uses a second list to store the sorted positions (not the values) of the data.

pcs6840288



The sorted list of values is referred to as List A, and the table of relative locations is List B. This is best illustrated with an example.

	LIST A		LIST B	
	Location	**Value in that Location**	**Location**	**Position in List A prior to sorting**
Before Sort	R0129	+02178	R0145	
	R0130	+31642	R0146	
	R0131	-24567	R0147	
	R0132	+18921	R0148	
	R0133	+24133	R0149	
After Sort	R0129	-24567	R0145	00003
	R0130	+02178	R0146	00001
	R0131	+18921	R0147	00004
	R0132	+24133	R0148	00005
	R0133	+31642	R0149	00002

The Sort function does not use a pointer. The first address specified is a data storage location.

Entering a Sort Function

The Sort function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), List Functions (F4), and then Sort Ascending (F5).

```
   R*****          R****** Const
-| LIST A  SORT  LIST B     LEN |-
```

3. The cursor is at LIST A. Using the numeric keypad, type in the beginning address of data to be sorted. It may be any available register. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to LIST B. Enter the beginning address for the Table B, which will store the relative positions of the data prior to the sort. This address may be any available register. After entering the reference, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the list. It must be a constant from 1 to 63. After entering the length, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

SECTION 10
Matrix Functions

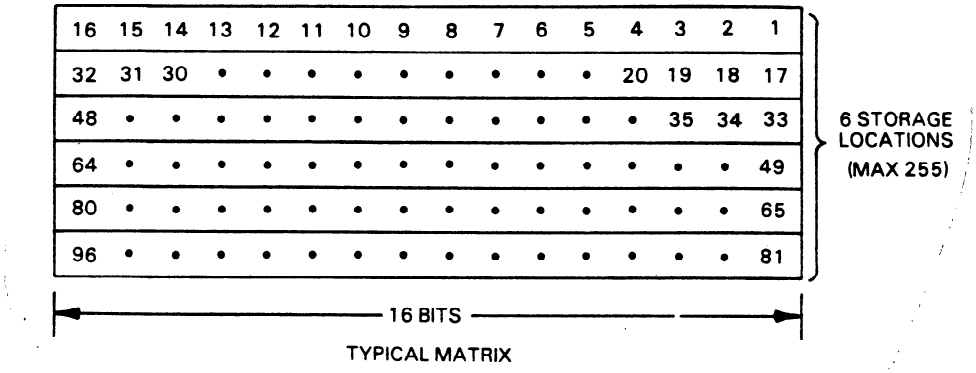
Use this section as a reference to the Matrix functions:

- Logical AND
- Logical IOR
- Logical EOR
- Logical Invert
- Matrix Compare
- Bit Clear
- Bit Set
- Bit Shift

A matrix consists of one or more adjacent 16-bit storage locations. Either registers or I/O can be assigned to matrixes.

Each bit in a matrix can be identified by its position - the first bit is considered to be number 1.

pcs6840291



A matrix may contain as many as 255 storage locations, each with 16 bits. Therefore, one matrix may contain from 16 to 4080 bits.

If a matrix is built in the I/O area, functions performed on the matrix will ignore the override status of any references contained in the matrix.

All matrixes are retentive during power failure or scan interruption.

Logical AND

The Logical AND function is useful to build masks or screens, where only certain bits are passed through (those that are opposite a 1 in the mask), and all other bits are set to 0. The Logical AND function can also be used to clear up to 255 registers by ANDing them with another matrix known to contain all 0s. Matrix structures can overlap. Each scan that power is received, the Logical AND function examines each bit in Matrix A and the corresponding bit in Matrix B, beginning at the first (lowest addressed) bit in each. For each two bits examined, if both are 1, then a 1 is placed in Matrix C. If either or both of the bits is a 0, then a 0 is placed in Matrix C in that location.

Bit in Matrix A	AND	Bit in Matrix B	Result in Matrix C
0		0	0
0		1	0
1		0	0
1		1	1

The Logical AND function passes power flow to the right only when Matrix C is all 0s. No corresponding bits in Matrix A and Matrix B are 1 at the same time.

Entering a Logical AND Function

The Logical AND function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function.
2. Select Advanced Mnemonic Group (F7), Matrix Functions (F5), and then Logical AND (F1).

```
 ******* ******* ******** Const
-|  A  AND  B   =    C     LEN |
                           ***
```

3. The cursor is at A. Type in the beginning address for Matrix A. It may be any available register or I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the beginning address for Matrix B. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Using the same parameters, enter the beginning address for Matrix C. Then, press CTRL-E (or the Enter key).
6. The cursor moves to LEN. Enter the length of the three matrixes. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
7. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Logical IOR

The Logical IOR function is useful to combine matrixes, and to control many outputs through the use of one simple logical structure. The Logical IOR is the equivalent of two relay contacts in parallel, multiplied by the number of bits in the matrix. It can be used to drive indicator lamps directly from the input status, or superimpose blinking conditions on status lights. Each scan that power is received, the Logical IOR function examines each bit in Matrix A and the corresponding bit in Matrix B, beginning at the first (lowest addressed) bit in each. For each two bits examined, if either or both are 1, then a 1 is placed in Matrix C in that location.

Bit in Matrix A	IOR	Bit in Matrix B	Result in Matrix C
0		0	0
0		1	1
1		0	1
1		1	1

The Logical IOR function passes power flow whenever power is received.

Entering a Logical IOR Function

The Logical IOR function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Matrix Functions (F5), and then Logical Inclusive OR (F2).

```
 ******* ******* ******** Const
-|  A  IOR  B   =    C     LEN |
                           ***
```

3. The cursor is at A. Using the numeric keypad, type in the beginning address for Matrix A. It may be any available register or I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the beginning address for Matrix B. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Using the same parameters, enter the beginning address for Matrix C. Then, press CTRL-E (or the Enter key).
6. The cursor moves to LEN. Enter the length of the three matrixes. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
7. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Logical EOR

The Logical EOR function is useful to quickly compare two matrixes, or to blink a group of bits at the rate of one ON state per two scans. Each scan that power is received, the Logical EOR function examines each bit in Matrix A and the corresponding bit in Matrix B, beginning at the first (lowest addressed) bit in each. For each two bits examined, if either one, but not both, is 1, then a 1 is placed in Matrix C in that location. If Matrix B and Matrix C begin at the same reference, a 1 placed in Matrix A will cause the corresponding bit in Matrix B to alternate between 1 and 0, changing state with each scan as long as power is received. Longer cycles can be programmed by pulsing the power flow at twice the desired rate of flashing; the power flow pulse should be one scan long (one-shot coil or self-resetting timer).

Bit in Matrix A	AND	Bit in Matrix B	Result in Matrix C
0		0	0
0		1	1
1		0	1
1		1	0

The Logical EOR function passes power flow only when all bits in Matrix C are zeros, indicating that all bits in Matrix B are the same as all bits in Matrix A.

Entering a Logical EOR Function

The Logical EOR function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function.
2. Select Advanced Mnemonic Group (F7), Matrix Functions (F5), and then Logical Exclusive OR (F3).

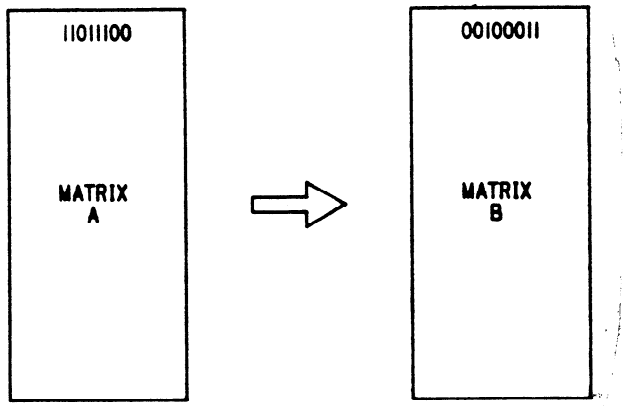
```
 ******* ******* ******** Const
-|  A  EOR  B   =    C     LEN |
                           ***
```

3. The cursor is at A. Using the numeric keypad, type in the beginning address for Matrix A. It may be any available register, or an I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the beginning address for Matrix B. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Using the same parameters, enter the beginning address for Matrix C. Then, press CTRL-E (or the Enter key).
6. The cursor moves to LEN. Enter the length of the three matrixes. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
7. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Logical Invert

Each scan that power is received, the Logical Invert function sets each bit in Matrix B to the opposite of the state of the corresponding bit in Matrix A.

a40035



All bits are altered on each scan that power is received, making Matrix B a mirror image of Matrix A. The Logical Invert function passes power flow whenever power is received.

Entering a Logical Invert Function

The Logical Invert function can be placed in columns 1 to 7 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Matrix Functions (F5), and then Logical Invert (F4).

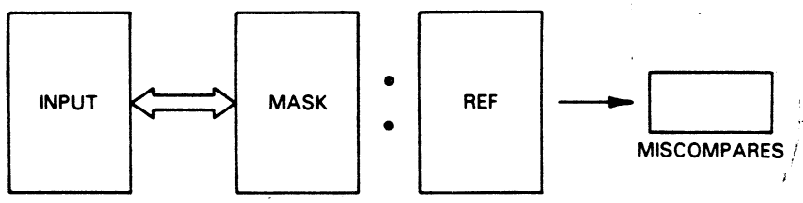
```
 ******* *******  Const
-|   A INV    B     LEN |
                    ***
```

3. The cursor is at A. Using the numeric keypad, type in the beginning address for Matrix A. It may be any available register, or an I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the beginning address for Matrix B. Then, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the three matrixes. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Matrix Compare

Each scan that power is received, the Matrix Compare function compares the bits in Matrix A with the corresponding bits in Matrix B. If a miscompare is located, the correspondingly-numbered bit in a third matrix, which serves as a mask, is checked. If the mask bit is a 1, the comparison continues until another miscompare is located, or until the end of the matrixes is reached. If the mask bit is a 0, the function places the bit number (its position in the matrixes) in a fault register, sets the mask bit to a 1, and passes power to the right.

pcs6840293



The following example represents the four possible cases. They are shown in matrix positions 1 through 4, but could occur anywhere, and in any combination, in the matrixes.

Bit #	Matrix A	Matrix B	Mask Bit	Action	Value in Fault Register
1	0	0	not checked		1
2	1	1	not checked		1
3	1	0	1	none	1
4	1	0	0	set mask bit to 1	4

The miscompare at bit #4 would stop the comparison until the next scan, because the mask bit is a 0. On the next scan when power is received, the compare would begin at bit location 4. Because the mask bit was now a 1, the function would continue to compare the matrixes until another miscompare was located.

Matrix A and Matrix B can represent any adjacent registers or I/O locations. For example, Matrix A might contain the status of outputs, such as solenoids or motor starters. Matrix B might contain their input state feedback, for example, limit switches or auxiliary contacts.

When the Matrix Compare function locates a miscompare between Matrix A and Matrix B, the function places the location of the bit (its position in the matrix) in a register. While the bit number is in the register, it can be accessed by other functions, such as the Source to Table Move function.

When power flows to the Matrix Compare function, the value in the fault number register increments by one. The result bit number is used as the bit location in the matrixes to start the comparison between A and B. The comparison continues and the location is indexed until a miscompare is located, or until the end of the matrixes is reached.

When the end of the matrixes is reached, there is no power flow and the fault location contains the length of the matrixes plus one.

When the compare begins, if the value in the fault register is equal to or greater than the matrix length, the fault register is reset to 1. Therefore, if no miscompares are located and the compare starts at the beginning of the matrix, it can evaluate all bits in the matrix in each scan.

All matrixes can be changed or reloaded using the functions described in this chapter. For example, the mask matrix can be reset to all zeros by programming a Logical Exclusive OR with the mask matrix

specified as both Matrix A and Matrix B. The input and reference matrixes can be reloaded by programming a Logical Exclusive OR with the mask location specified as both Matrix A and Matrix B, then programming an Inclusive OR with the new status. Another way to reload the input and reference matrixes is to use an Extended Move function. The fault register can be loaded from a table (Table to Destination Move) to cause the comparison to start at a specific bit location. Many other applications can be programmed.

Entering a Matrix Compare Function

The Matrix Compare function can be placed in columns 1 to 4 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Matrix Functions (F5), and then Matrix Compare (F5).

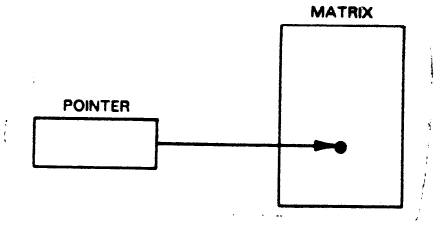
```
          ******* ******* ******* *******  Const
-|COMPARE INPUT     REF     MASK    FAULT     LEN|
                                             ***
```

3. The cursor is at INPUT. Using the numeric keypad, type in the beginning address for the input matrix. It may be any available register, or an I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to REF. Using the same parameters, enter the beginning address for the reference matrix. Then, press CTRL-E (or the Enter key).
5. The cursor moves to MASK. Using the same parameters, enter the beginning address for the mask matrix. Then, press CTRL-E (or the Enter key).
6. The cursor moves to FAULT. Using the same parameters, enter the address of the reference where the bit number of a fault will be stored. Then, press CTRL-E (or the Enter key).
7. The cursor moves to LEN. Enter the length of the three matrixes. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
8. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Bit Set/Sense

Every scan, the Bit Set function senses or sets a pointer-designated bit within a matrix. The number of the bit must be greater than 0 and within the total size in bits of the matrix (matrix length x 16).

pcs6840294



If the Bit Set function receives power flow from the left, it sets the designated bit to 1 and passes power to the right.

If the Bit Set function does not receive power flow from the left, it senses the state of the designated bit. If the state is 1, the Bit Set function generates power to the right.

If the pointer is out of range, the Bit Set function does not output power flow.

Entering a Bit Set Function

The Bit Set function can be placed in columns 1 to 6 of the first line of a rung.

1. Select Advanced Mnemonic Group (F7), Matrix Functions (F5), Bit Matrix Group (F6), and then Bit Set (F1).

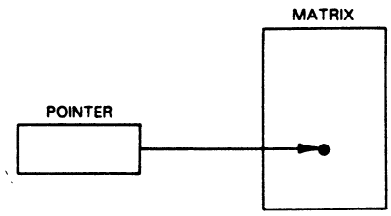
```
 *******          *******  Const
-|  BIT  SET     MATRIX       LEN |
                              ***
```

2. The cursor is at BIT. Using the numeric keypad, type in matrix position of the bit to be set/sensed. It may be any available register, an I/O reference that begins on a byte boundary, or a constant. The value must be in the range 1-4080. After entering the reference, press CTRL-E (or the Enter key).
3. The cursor moves to MATRIX. Enter the beginning address of the matrix where the bit is located. It may be any available register, or an I/O reference that begins on a byte boundary. Then, press CTRL-E (or the Enter key).
4. The cursor moves to LEN. Enter the length of the matrix. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Bit Clear/Sense

Every scan, the Bit Clear function senses or clears a pointer-designated bit within a matrix. The number of the bit must be greater than 0 and within the total size in bits of the matrix (matrix length x 16).

pcs6840294



If the Bit Clear function receives power flow from the left, it clears the designated bit to 0. It does not pass power to the right.

If the Bit Clear function does not receives power flow from the left, it senses the state of the designated bit. If the state is 1, the Bit Clear function generates power to the right.

If the pointer is out of range, the Bit Clear function outputs power flow, regardless of the input power flow.

Entering a Bit Clear Function

The Bit Clear function can be placed in columns 1 to 6 of the first line of a rung.

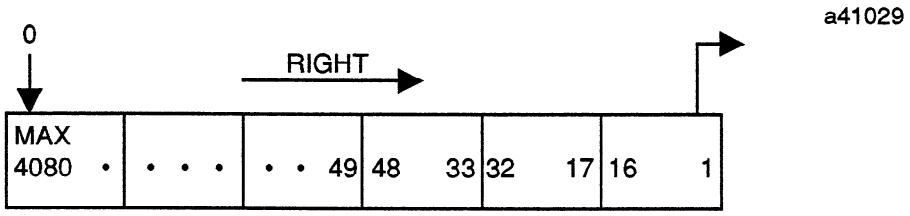
1. Select Advanced Mnemonic Group (F7), Matrix Functions (F5), Bit Matrix Group (F6), and then Bit Clear (F2).

```
 *******          *******  Const
-|  BIT  CLEAR  MATRIX       LEN |
                             ***
```

2. The cursor is at BIT. Using the numeric keypad, type in matrix position of the bit to be cleared/sensed. It may be any available register, an I/O reference that begins on a byte boundary, or a constant. The value must be in the range 1-4080. After entering the reference, press CTRL-E (or the Enter key).
3. The cursor moves to MATRIX. Enter the beginning address of the matrix where the bit is located. It may be any available register, or an I/O reference that begins on a byte boundary. Then, press CTRL-E (or the Enter key).
4. The cursor moves to LEN. Enter the length of the matrix. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Shift Right

Every scan that power is received, the Shift Right function shifts all the bits in a matrix a specified number of places to the right (that is, toward less-significant bit locations).



The number of places specified for the shift must be greater than 0 and within the total size in bits of the matrix (matrix length x 16). Otherwise, no shift occurs, and no power flow is generated.

When the shift occurs, the specified number of bits is shifted out of the matrix. Power flow is passed to the right only if the last bit that is shifted out of the matrix is a 1.

As bits are shifted out of the low end of the matrix, zeros are loaded in from the most significant bit of the last word in the matrix.

Entering a Shift Right Function

The Shift Right function can be placed in columns 1 to 6 of the first line of a rung.

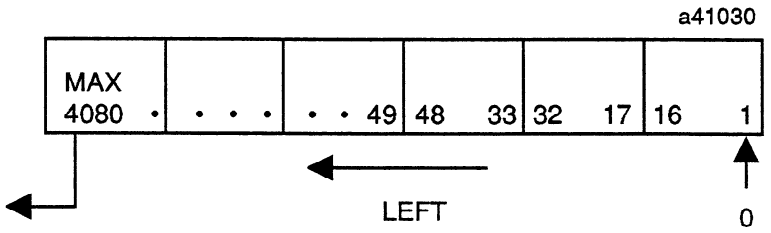
1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Matrix Functions (F5), Bit Matrix Group (F6), and then Bit Shift Right (F4).

```
          ******* *******  Const
-| SHIFT RT   N   MATRIX    LEN |
                            ***
```

3. The cursor is at RT N. Using the numeric keypad, type in number of places for the bits to be shifted. It may be any available register, an I/O reference that begins on a byte boundary, or a constant. The value of each must be in the range 1-4080. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to MATRIX. Enter the beginning address of the matrix where the bits will be shifted. It may be any available register, or an I/O reference that begins on a byte boundary. Then, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the matrix. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Shift Left

Every scan that power is received, the Shift Left function shifts all the bits in a matrix a specified number of places to the left (that is, toward more-significant bit locations).



The number of places specified for the shift must be greater than 0 and within the total size in bits of the matrix (matrix length x 16). Otherwise, no shift occurs, and no power flow is generated. When the shift occurs, the specified number of bits is shifted out of the matrix. Power flow is passed to the right only if the last bit that is shifted out of the matrix is a 1. As bits are shifted out of the high end of the matrix, zeros are loaded in from the least significant bit of the first word in the matrix.

Entering a Shift Left Function

The Shift Right function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Matrix Functions (F5), Bit Matrix Group (F6), and then Bit Shift Left (F3).

```
          ******* *******  Const
-| SHIFT LEFT N   MATRIX    LEN |-
                            ***
```

3. The cursor is at LEFT N. Using the numeric keypad, type in number of places for the bits to be shifted. It may be any available register, an I/O reference that begins on a byte boundary, or a constant. The value of each must be in the range 1-4080. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to MATRIX. Enter the beginning address of the matrix where the bits will be shifted. It may be any available register, or an I/O reference that begins on a byte boundary. Then, press CTRL-E (or the Enter key).
5. The cursor moves to LEN. Enter the length of the matrix. It must be a constant from 1 to 255. After entering the length, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

SECTION 11
Control Functions

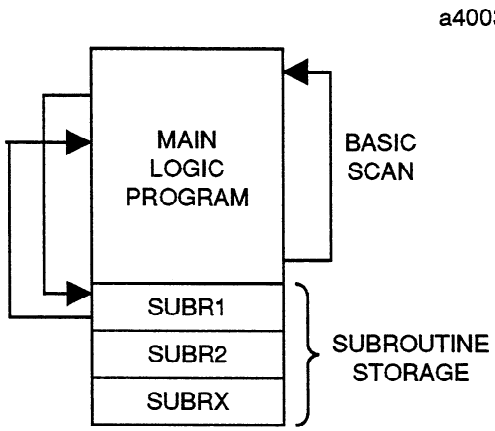
The Control Functions are used to alter or control the normal sequence of logic operations.

Use this section as a reference to the Control functions:

- Do Subroutine
- Return from Subroutine
- Suspend I/O
- Do I/O
- Status

Do Subroutine

Each scan that power is received, the Do Subroutine function causes the scan to go immediately to the designated subroutine and execute it. After the subroutine has executed the number of times specified by the Do Subroutine function, control returns to the main body of the program.



A subroutine is a segment of ladder logic. It is just like the rest of the program, except that it cannot contain a Do Subroutine function. In addition, timers may not function as expected, if located within a subroutine.

As many as 16 subroutines can be created, and stored directly after the main program.

Main Ladder Diagram Program
END-OF-SWEEP
Subroutine 1
RETURN
Subroutine 2
RETURN
↓
Subroutine 16
RETURN
END-OF-SWEEP
END-OF-SWEEP

A single End of Sweep function at the end of the main program separates it from any subroutines that follow. Each subroutine ends with a Return function. The system places two End of Sweep functions together at the end of the entire program.

The Return function causes the scan to return to the main program, and power flow passes to the right of the Do Subroutine function.

Because subroutines are located outside the main program, they are not scanned routinely. A subroutine is scanned only if called by a Do Subroutine function, or by the use of an interrupt. There is no limit to the number of Do Subroutine functions that may be included in a program.

Note that an excessive number of subroutine calls and execution cycles may cause the WatchDog Timer to exceed its range and shut down the CPU.

Calling Subroutines with Interrupts

Instead of using a Do Subroutine function to call a subroutine, it is possible to use interrupt inputs in the hardware I/O section. When an interrupt input is turned on, the main program stops. Control jumps immediately to the subroutine associated with the interrupt. The subroutine then executes once.

There are 8 input interrupts in the Main I/O chain, and 8 interrupts in the Auxiliary I/O chain. These interrupts are triggered by the Interrupt Input Module, as described in the *Series Six PLC Installation and Maintenance Manual* (GEK-25361) and the *Application Guide for the Series Six PLC Programmable Controller*. (A brief description of programming for the Interrupt Input Module is given in chapter 12 of this manual).

The first 8 interrupts use I1001 to I1008. They are located in a special DC input card in the main chain. The second 8 interrupts use addresses AI1001 to AI1008. They are triggered by the same type of card in the Auxiliary I/O chain. Each interrupt is associated with one of the 16 subroutine numbers. In addition, each has a fixed priority of service. Interrupts occur on transition of the input signal.

Address	Priority	Associated Subroutine	
I1001	1th	1	
I1002	2th	2	
I1003	3th	3	
I1004	4th	4	Main
I1005	5th	5	I/O Chain
I1006	6th	6	
I1007	7th	7	
I1008	8th	8	
AI1001	9th	9	
AI1002	10th	10	
AI1003	11th	11	
AI1004	12th	12	Auxiliary
AI1005	13th	13	I/O Chain
AI1006	14th	14	
AI1007	15th	15	
AI1008	16th	16	

Entering a Do Subroutine Function

Enter a Do Subroutine function at each location in the main program where control should jump to the subroutine.

The Do Subroutine function can be placed in columns 1 to 7 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Control Functions (F6), and then Do Subroutine (F1).

```
            Const   *******
-| DO SUB     N      REPS |
             ***
```

3. The cursor is at N. Using the numeric keypad, type in number of the subroutine. It must be a constant from 1 to 16. After entering the number, press CTRL-E (or the Enter key).
4. The cursor moves to REPS. Enter the reference that contains the number of times the subroutine execution will be repeated before the Return function at the end of the subroutine is executed, and control returns to the main program.

   The content of the register that controls the number of repetitions may be changed by the program, but must always be within the range 1 to 255. If the reference contains the value zero (0) when power is received by the function, the subroutine will not execute. If the reference contains the value 1, the subroutine will execute once. The reference may be any available register, or an I/O reference that begins on a byte boundary. After entering the reference, press CTRL-E (or the Enter key).

   NOTE

   The number of executions is stored in the low order half of the reference. The high order half serves as a counter during subroutine execution. After each execution, the number in the high order half increments by one. When this number equals the value in the low order half of the reference, the high order portion is cleared to zero and the Return function is executed.

5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Return

A Return function must always be located at the end of a subroutine, and may never be included in a main program.

Each time the subroutine executes, power flows to the Return function. The Return function then increments by one the value in the high order half of the reference containing the number of subroutine executions (refer to the Do Subroutine function).

If the value in the high order half of the reference is less than the value in the low order half (which is the number of intended subroutine executions), control is passed back to the beginning of the subroutine, which executes again.

If the value in the high order half of the reference equals or exceeds the value in the low order half, control returns to the main program at a point directly after the Do Subroutine function that caused the exit.

Entering a Return Function

Enter a Return function at the end of a subroutine.

The Return function can be entered only in column 1 of a rung.

1. Select Advanced Mnemonic Group (F7), Control Functions (F6), and then Return from Subroutine (F2).

```
-|   Return     |
```

2. Press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Suspend I/O

Each scan that power is received, the Suspend I/O function causes the CPU to skip all I/O servicing at the end of that scan. That means all output devices are held in their current state. The input table is not updated with data from inputs, unless the program contains active Do I/O functions.

If the program contains more than one Suspend I/O functions, only one need be active for I/O service to be suspended.

The Suspend I/O function passes power to the right on each scan that the function is active.

Entering a Suspend I/O Function

The Suspend I/O function can be placed in columns 1 to 8 of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Control Functions (F6), and then Suspend I/O (F3).

```
-| SUSPEND I/O |
```

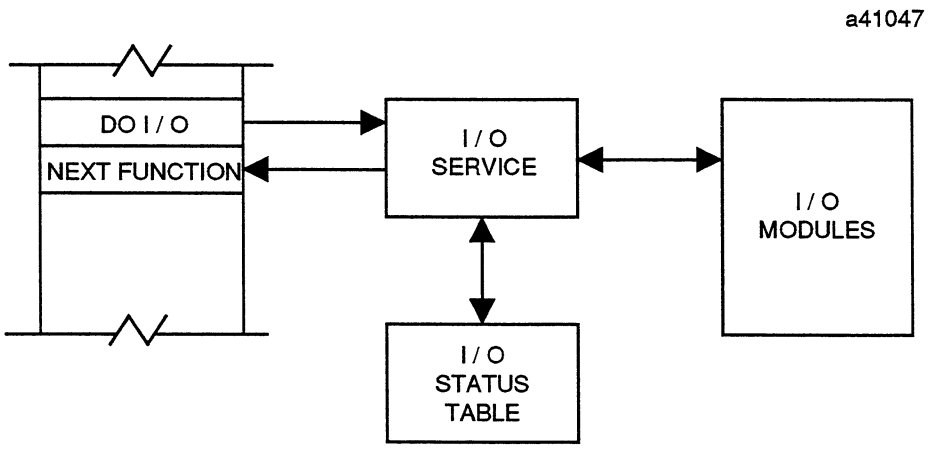
3. This function has no references. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Do I/O

Each scan that power is received, the Do I/O function causes the CPU to service selected I/O. This is in addition to the I/O servicing at the end of the scan.

The Do I/O function allows the most recent values of the references to be obtained for program logic, and updates references for external use. It can also be used to rapidly service just the I/O pertinent to the program. It can be used in conjunction with the Suspend I/O function, which eliminates the I/O servicing at the end of a scan.

The I/O is serviced in full bytes. The addresses for the I/O to be serviced are stored in program references. It is these program references that are used to specify the servicing range, not the actual module addresses.

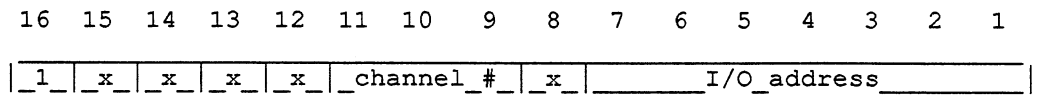


When the Do I/O function receives power, the values entered as the starting and ending references are compared. If the start reference is less than or equal to the end reference, and both values are between 1 and 1000, the I/O, whose addresses are contained in the references (but never less than 8 points), are serviced and power flow is passed to the right.

If the reference start is greater than the reference end, or outside the range 1 to 1000, the I/O are not serviced, and power is not passed to the right.

Using the DO I/O Function to Address 16K Inputs and Outputs

The DO I/O function can be used to address the full 16K inputs and 16K outputs of the expanded I/O references. When expanded I/O is selected, the bits in the Do I/O function references have the meanings shown below. These can be placed in the references using an A to B Move or similar function.



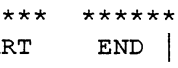
Bit 16 must be set to 1 by the program.
If a constant value is used, bits 15 and 16 must both be set to 1.
Bits 9, 10, and 11 are the channel number, 0 - 7.
Bits 1 through 7 specify the I/O address.

The Do I/O instruction will scan all inputs and outputs on both the Main and Auxiliary I/O chains, beginning with the specified (start byte, channel) and continuing through the (end byte, channel).

Entering a Do I/O Function

The Do I/O function can be placed in columns 1 to 7 of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Control Functions (F6), and then Do I/O (F4).



3. The cursor is at START. Using the decimal keypad, type in the reference that contains the beginning I/O address. It may be a constant from 1 to 1000, any available register, or an I/O reference. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to END. Using the same parameters, enter the reference that contains the final address. Then, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

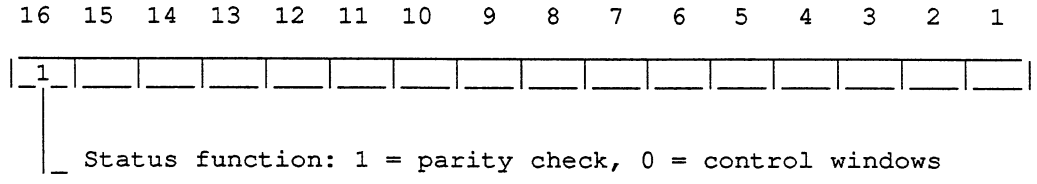
Status

The Status function can be used to control the communications windows between the CPU and external devices such as the ASCII I/O, DPU, CCM, and computer interface. There are two types of communications windows: the executive window which is executed once each CPU scan, and individual windows controlled by the SCREQ and DPREQ functions (two of the Basic functions).

The Status function can be used to turn communications with the CCM or DPU and ASCII/BASIC module off or on. If turned off, both the executive window and individual communications requests to a device are disabled. For example, if the status of the CCM window is set to 1 (disable), no executive window or SCREQ communications with the CCM will occur until the status is re-enabled. This function can help control scan time in certain critical applications such as those where a guaranteed minimum response time to an interrupt is required.

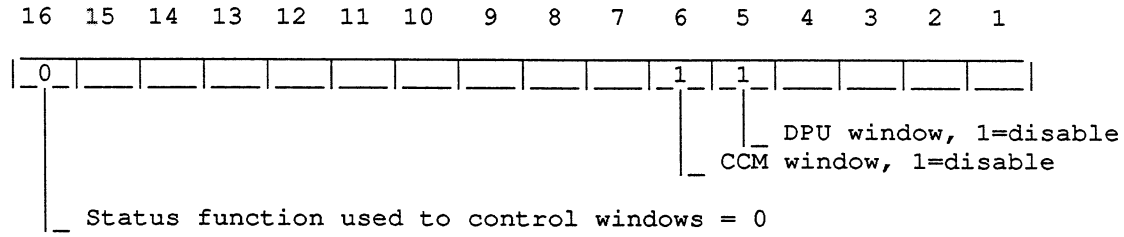
On CPUs having version 105 software or later, the Status function can alternatively be used to control the number of data transmission retries that will be made after a parity error is detected.

The Status function uses a single 16-bit reference. In CPUs with version 105 or later software, bit 16 selects control of windows or parity error retries. If the CPU does not have version 105 or later software, bit 16 is not used.



Using the Status Function to Control the Windows

When used to control the communications windows, the bits in the Status function reference have the meanings shown below.



Bit 5 controls the DPU window and DPREQ commands. Bit 6 controls the CCM window and SCREQ commands. If either bit is equal to 1, that window is disabled. If a bit is = 0, the window is generated normally.

When the Status function does not receive power flow, bit 5 and bit 6 reflect the status of the windows. If one of these windows is available, when the proper hardware and connections are in place and turned on, the bit is on (1). When the Status function receives power flow, the direction of data transfer is reversed. The bits are taken from the reference and used to turn the window on or off. When the function is active, power flows to the right.

Power In	Function	Bit 5	Bit 6	Means
no	sense	0 1	0 1	window is enabled window is disabled
yes	control	0 1	0 1	enable window disable window

If a 1 is placed in the reference at either bit location, data communication with that device stops immediately. As long as the 1 remains in the bit location, no data transfer is possible. To restore normal communication, a 0 can be placed in the bit location, or power flow to the Status function can be removed.

More than one Status function can be used in a program. The last one encountered during the scan will control window execution.

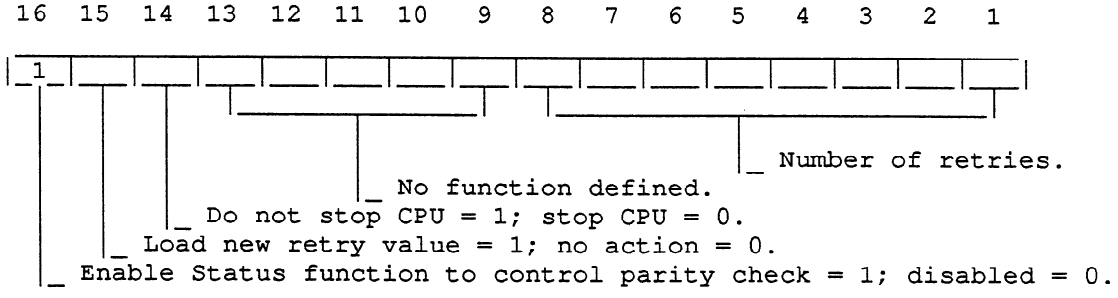
If you develop a program with the Advanced, Expanded or Expanded II function set selected in the Scratch Pad, but the program will be used by a programmable controller CPU that has the Extended function set (version #101-103), *do not use the Status function to close any communications window.* Doing so causes an error, and the CPU will not re-open connunications until the error is cleared.

Using the Status Function to Control Parity Check Retries

In CPUs using version 105 or later software, the Status function can be used to specify and control the number of retries when a parity fault is encountered. Normally, when the CPU services the I/O, it executes a parity check on each 8-bit byte of sent or received data. If an error is detected, the transfer is repeated. A second failure in the same transfer indicates a major I/O problem and causes the CPU to stop scanning.

It is recommended that if the Status function is used to control parity checking, it be located in the beginning of the program if interrupts inputs are used. In that way, it will be solved before the interrupts service a subroutine.

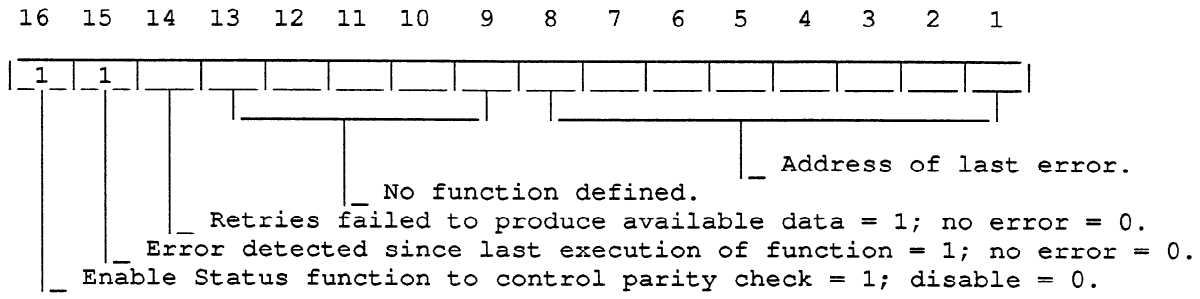
If bit 16 of the Status function reference is 1, the function controls the number of parity check repeats that will be performed on transmitted data. When power flows to this function, the bits have the following meanings:



Bit 16 controls how the Status function reference is used. Bits 1 to 8 store a number of retries. If bit 15 is set to 1, this number of retries is loaded and used. Bit 14 controls whether the CPU will be stopped if the parity check fails.

If a parity fault is detected, the same location will be serviced the number of retries in bits 1 through 8 before an error is determined. If the retries indicate faulty parity, bit 14 controls whether the CPU is stopped or continues at the next I/O point.

When power does not flow to this function, the bits have the following meanings:



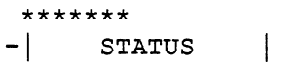
Bit 15 is 1 if at least one parity error was encountered since the last time the function was executed. It is set even if a retry was successful. Bit 14 is set only if the specified number of retries failed to produce available data. The lower 8 bits store the last I/O address or memory location where an error was encountered.

If an error occurs that is not corrected in the specified number of retries, all I/O status for that address (including auxiliary I/O) is not updated or changed on that scan.

Entering a Status Function

The Status function can be placed in columns 1 to 8 of any line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Control Functions (F6), and then Status (F5).



3. The cursor is at the left. Using the numeric keypad, type in number of the reference. It may be any available register, or an I/O reference beginning on a byte boundary. Then, press CTRL-E (or the Enter key).
4. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Chapter 14
Expanded Functions

GEK-25379

Use this chapter as a reference to the Expanded and Expanded II program functions. Within the chapter, functions are grouped into sections. After a description of the function, you will find step-by-step instructions for entering the function in a rung.

To run a program using the Expanded II function set, the programmable controller CPU must have an Arithmetic Control Unit Module (IC600CB524). If this module is not in the CPU, the Expanded II instruction set should not be used.

For these functions, all I/O and register references may be used if available in the CPU.

Floating Point Arithmetic Functions include:

- Floating Point Addition
- Floating Point Subtraction
- Floating Point Multiplication
- Floating Point Division
- Floating Point Greater Than
- Integer to Floating Point Conversion
- Floating Point to Integer Conversion

Control Functions include:

The Expanded function set provides three Control functions:

- CPU Configuration Function
- Window Function
- Status Function

The Expanded II function set provides an additional function for use with Series 90-70 I/O racks:

- 90-70 I/O Rack Service

Expanded Arithmetic Functions include:

- Expanded Compare

Configuring the Scratch Pad

Before programming, you should configure the Scratch Pad to match the capabilities of the CPU that will receive the program. The Expanded function set must be selected to use any of the functions described in this chapter. The Expanded II function set must be selected to use the 90-70 I/O Rack Service instruction.

Also, specify the amount of CPU register memory that is available. After the Scratch Pad selections are made, the Logicmaster 6 software will prevent the entry of references that are incompatible with the function level and register memory of the CPU.

SECTION 1
Floating Point Arithmetic Functions

Use this section as a reference to the Floating Point Arithmetic functions:

- Floating Point Addition
- Floating Point Subtraction
- Floating Point Multiplication
- Floating Point Division
- Floating Point Greater Than
- Integer to Floating Point Conversion
- Floating Point to Integer Conversion

All of these functions are reached by pressing the Advanced Functions key.

Entering Floating Point Values

To enter floating point numbers:

1. Set the work area numeric line to floating point format by pressing ALT-C/DP (or the ALT and keypad + keys). The numeric line is divided into two parts, left for the mantissa and right for the exponent of the floating point number. The cursor begins at the mantissa field: [+2.81660] -42.
2. Enter the mantissa.
3. Press CTRL-S (or the Select key) to move cursor to the exponent field: +2.81660 [-42].
4. Enter the exponent. If none is entered, the system assumes an exponent of +00.

Entering a Floating Point Constant

To enter a floating point constant:

1. Enter a constant on the reference line.
2. Press CTRL-S (or the Select key) to move the work area cursor to the numeric line. If the numeric line is not already in floating point format, press ALT-C/DP (or the ALT and keypad + keys).
3. Enter the value, as explained above.

Floating Point Addition

The Floating Point Addition function adds a floating point value in one reference to the floating point value in another reference and places the result in a third reference. All three references require two 16-bit words.

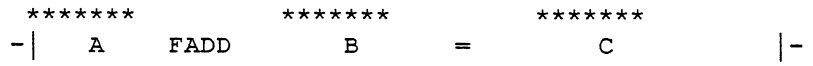
Every scan that power is received, the FADD function calculates the result using floating point mathematics. The result is placed in reference C. Only the content of reference C is altered by this function.

The function outputs power flow if the result is greater than the capacity of floating point storage, or if references A and B are opposite signed infinities, or if either reference is not a number (invalid format).

Entering a Floating Point Addition Function

A Floating Point Addition function can be placed in columns 1 to 4 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), Floating Point Arithmetic (F6), and then Floating Point Addition (F1).



3. The cursor is at A. Using the numeric keypad, type in the reference of the first value to be added. It may be any available register or I/O reference, or a constant. The range for floating point values is $\pm(1.401298 \times 10^{-45}$ to $3.402823 \times 10^{+38})$. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the reference for the second value to be added. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. It may be any available register or I/O reference. After entering the reference, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Floating Point Subtraction

The Floating Point Subtraction function subtracts a floating point value in one reference from the floating point value in another reference and places the result in a third reference. All three references require two 16-bit words.

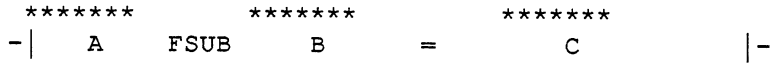
Every scan that power is received, the FSUB function calculates the result using floating point arithmetic. The result is placed in reference C. Only the content of reference C is altered by this function.

The function outputs power flow if the result is greater than the capacity of floating point storage, or if references A and B are infinities having the same sign, or if either reference A or B is not a number (invalid format).

Entering a Floating Point Subtraction Function

A Floating Point Subtraction function can be placed in columns 1 to 4 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), Floating Point Arithmetic (F6), and then Floating Point Subtraction (F2).



3. The cursor is at A. Using the numeric keypad, type in the reference of the first value to be added. It may be any available register or I/O reference, or a constant. The range for floating point values is $\pm(1.401298 \times 10^{-45}$ to $3.402823 \times 10^{+38})$. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the reference for the second value to be added. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. It may be any available register or I/O reference. After entering the reference, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Floating Point Multiplication

The Floating Point Multiplication function multiplies the value in one reference by the value in another reference and place the result in a third reference. All three references require two 16-bit words.

Every scan that power flows to the function, the system multiplies the value in reference A by the value in reference B and places the signed result in reference C. The possible sign of the result is shown below.

Reference A	Reference B	Reference C
+	+	+
+	-	-
-	+	-
-	-	+

The Floating Point Multiplication function outputs power flow only when the result is too large or too small to store in floating point format, either reference A or B is an infinity and the other is zero, or either reference A or B is not a number (invalid format).

Entering a Floating Point Multiplication Function

A Floating Point Multiplication function can be placed in columns 1 to 4 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), Floating Point Arithmetic (F6), and then Floating Point Multiplication (F4).

```
 *******          *******          *******
-|    A    FMULT      B        =     C        |-
```

3. The cursor is at A. Using the numeric keypad, type in reference of the first value. It may be any available register or I/O reference, or a constant. The available range for floating point values is $\pm(1.401298 \times 10^{-45}$ to $3.402823 \times 10^{+38})$. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the reference for the value to be multiplied by reference A. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Enter the reference where the result of the multiplication will be placed. It may be any available register or I/O reference. After entering the reference, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Floating Point Division

The Floating Point Division function divides the value in one reference by the value in another reference and place the result in a third reference. All three references require two consecutive 16-bit storage locations.

The possible signs are shown below.

Reference A	Reference B	Quotient
+	+	+
+	-	-
-	+	-
-	-	+

The Floating Point Division function outputs power flow only when the result is too large or too small to store in floating point format, when reference B is zero, when both references A and B are infinities, or when either reference A or B is not a number (invalid format).

Entering a Floating Point Division Function

A Floating Point Division function can be placed in columns 1 to 4 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), Floating Point Arithmetic (F6), and then Floating Point Division (F4).

```
 *******          *******          *******
-|   A    FDIV       B        =     C        |-
```

3. The cursor is at A. Using the numeric keypad, type in reference of the first value. It may be any available register or I/O reference, or a constant. The range for floating point values is $\pm(1.401298 \times 10^{-45}$ to $3.402823 \times 10^{+38})$. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the reference for the value to be divided into reference A. Then, press CTRL-E (or the Enter key).
5. The cursor moves to C. Enter the reference where the result of the division will be placed. It may be any available register or I/O reference. After entering the reference, press CTRL-E (or the Enter key).
6. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Floating Point Greater Than

The Floating Point Greater Than function compares the value in one reference to the value in another reference. Both references require two consecutive 16-bit storage locations.

Every scan that power is received, the Greater Than function compares the content of reference A to the content of reference B. Comparison is based on the signed, floating point values of the contents.

Power flows only if the first value is greater than the second. Some example comparisons are shown below.

Reference A	Reference B	Power Flow ?
+2.34595+1	+2.34595+0	yes
+2.34595-1	+2.34595+1	no
+0.00000+0	+9.99999-1	no
+0.00000+0	+1.00000-1	no
+0.00000+0	-1.00000+1	no

Entering a Floating Point Greater Than Function

A Floating Point Greater Than function can be placed in columns 1 to 5 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), Floating Point Arithmetic (F6), and then Floating Point Compare (F5).

```
 *******                  *******
-|  A    FP GREATER THAN   B            |-
```

3. The cursor is at A. Using the numeric keypad, type in reference of the first value. It may be any available register or I/O reference, or a constant. The range for floating point values is $\pm(1.401298 \times 10^{-45}$ to $3.402823 \times 10^{+38})$. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the reference for the value to be divided into reference A. Then, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Convert Integer To Floating Point

Every scan that power is received, the Integer to Floating Point function reads the integer value of reference A and places the floating point equivalent in reference B. Both A and B require 16-bit words.

Power flow is generated whenever power is received.

Entering an Integer to Floating Point Function

An Integer to Floating Point function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), Floating Point Arithmetic (F6), and Integer/Floating Point Convert (F6).

```
 *******          *******
-| INTEGER  TO  FLOATING POINT |-
```

3. The cursor is at INTEGER. Using the numeric keypad, type in the reference of the first value. It may be any available register or I/O reference, or a constant. Then, press CTRL-E (or the Enter key).
4. The cursor moves to FLOATING POINT. Using the same parameters, enter the reference for the converted data. It may be any available register or I/O reference. Then, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Convert Floating Point to Integer

Every scan that power is received, the Floating Point to Integer function reads the floating point value of reference A and places the integer equivalent in reference B. Both references require 16-bit words.

Fractions are rounded up if the fraction is equal to or greater than .5 (i.e., 1.5 is rounded up to 2). If the fraction is less than .5 (i.e., 1.49999), it is rounded down to 1.

Power flow is generated if either:

- The floating point number to be converted is beyond the allowed range of double precision integers.
- The floating point number is not a valid floating point number.

Entering a Floating Point to Integer Function

A Floating Point to Integer function can be placed in columns 1 to 6 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), Floating Point Arithmetic (F6), and Floating Point/Integer Convert (F7).

```
 *******                 *******
-| FLOATING POINT  TO  INTEGER |-
```

3. The cursor is at FLOAT. Using the numeric keypad, type in reference of the first value. It may be any available register or I/O reference, or a constant. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to INTEGER. Using the same parameters, enter the reference for the converted data. Then, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

SECTION 2
Control Functions

The Expanded function set provides two Control Functions:

- CPU Configure
- Window
- Status

The Expanded II function set provides an additional function for use with Series 90-70 I/O racks:

- 90-70 I/O Rack Service

CPU Configuration

The CPU Configuration function must be present in the program for access to the CPU Configure screens, which are described in chapter 10, *Expanded Functions Menu.* The entries on the CPU Configure screens, including Expanded I/O scan, Bus Controller locations, Genius I/O Diagnostics, and other information, are stored with the program. Once stored, they become part of the CPU Configuration function, although they do not appear in the ladder logic.

Entering the Configuration Function

The CPU Configuration function must be located at column 1 of the first rung of a program. After this function is placed in the program, no other logic can be inserted before it. To enter the Configuration function at the start of a program:

1. In Edit mode, with the cursor located at column 1 of line 1, press Insert Rung (F5), Advanced Mnemonic Group (F7), Control Functions (F6), Configure (F6), and CPU Configure (F1).

```
-|SERIES SIX CONFIGURATION DATA|-
```

2. Press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

NOTE

If changes are made to the Expanded Functions screens after a program has been stored to the Series Six CPU, the CPU Configuration data in the stored program will not reflect the changes. The program must be stored to the CPU again, to include the new Expanded functions data.

Series 90-70 I/O Rack Service

The Series 90-70 I/O Rack Service function must be present in the program for access to the Series 90-70 I/O Rack Configuration and Display screens, which are described in chapter 10, *Expanded Functions Menu.*

The entries on the 90-70 I/O Rack Configure screen, including the rack number and configuration parameters and output default states for that rack, are stored with the program. Once stored, they become part of the Series 90-70 I/O Rack Service function, although they do not appear in the ladder logic.

This function should be placed in the program for each Series 90-70 I/O rack to receive a full I/O scan. When the CPU executes this function in the ladder logic, all I/O in the specified rack will be serviced. If a partial scan is preferred, use the DPREQ or Window instruction instead. For more information, refer to the *Series 90-70 I/O Application Guide* (GFK-0152).

Entering the Series 90-70 I/O Rack Service Function

To enter the 90-70 I/O Rack Service function:

1. In Edit mode, press Insert Rung (F5), Advanced Mnemonic Group (F7), Control Functions (F6), Configure (F6), and then 90-70 Configure (F2).

```
                    Const
-| 90-70  I/O       RACK    SERVICE |-
                    ***
```

2. Using the decimal keypad, enter the number of the rack to be serviced, (0 to 15). Then, press CTRL-E (or the Enter key).
3. If the rung is complete, press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

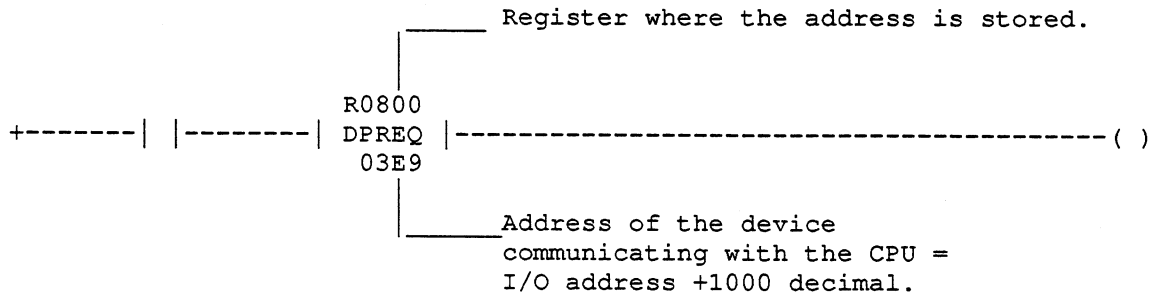
NOTE

If changes are made to the Series 90-70 Configuration screen (as described in chapter 10) after a program has been stored to the Series Six CPU, the stored program will not reflect the changes. The program must be stored to the CPU again to use the new selections.

Window

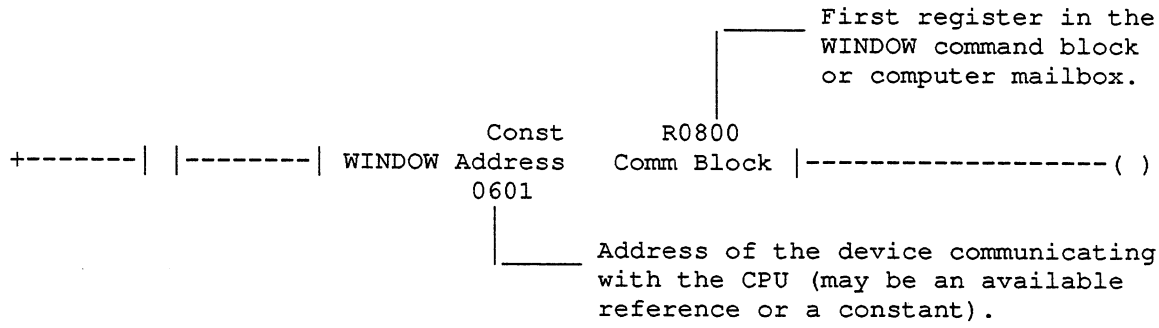
The Window function is a communications request that supports Expanded I/O addressing. It is similar to the DPREQ function used with Non-Expanded addressing. A DPREQ cannot select an Expanded I/O channel, while a Window function can.

The following two lines of logic show the difference between entering a DPREQ and a Window function:



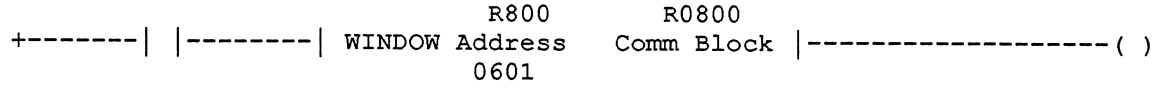
In the example DPREQ above, the function's register reference is R800 and is set to access a module at I/O address 0001. R800 stores the value 03E9. Other communications parameters for this DPREQ would be specified using the six registers immediately following R800 (that is, R801-R806). R800-R806 used in this example are referred to as the DPREQ command block.

If a Window function was used to replace this DPREQ, it might look like this:



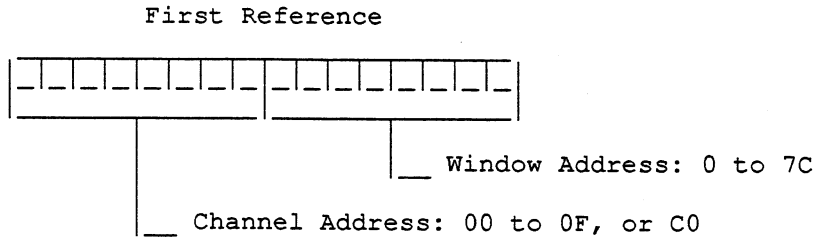
In the Window function, the address of the communicating device is given as the value for the Window Address reference. Other communications parameters are specified using registers beginning at the location given for the WINDOW command block. This is R800, on the right in the example above.

The WINDOW command block and the DPREQ command block differ only in the definition of the first register. The first register of the WINDOW command block should be the WINDOW device address. Thus, the example above could also be entered as:



Format for Entering the Window Address

Each scan that power is received, the Window function opens a window to the Expanded I/O system. The address is specified by the hexadecimal value stored in the lower byte of the first reference.



The Channel Address number stored in the high-order byte may be either:

00-0F: Available individual channel numbers. Individual channel numbers are determined by the DIP switch settings on the I/O transmitter modules. When Expanded addressing mode has been selected (jumper J2 over pins 2 and 3), the channel number for the card is determined by the first three switches on the backplane DIP switch pack. DIP switch positions for each channel of I/O are shown below. In the chart, x means the switch is in the open position (closed to the left). In the Window function, enter the hex value that corresponds to the card's channel number.

Main Chain		Auxiliary Chain		Switch Position		
Channel	**hex**	**Channel**	**hex**	**3**	**2**	**1**
0	00	8	08			
1	01	9	09			x
2	02	A	0A		x	
3	03	B	0B		x	x
4	04	C	0C	x		
5	05	D	0D	x		x
6	06	E	0E	x	x	
7	07	F	0F	x	x	x

CO: Broadcast channel number. C0 is converted to 80 by the CPU before opening the window. C0 is required, because 80 can't be stored in the upper byte of the reference.

The low-order byte of the Window function stores the module's card address. It may be any address from 0 to 7C (hex). The value 7D is reserved for the Interrupt Input Board. 7E and 7F are reserved for the DPU and PDT windows, and are not allowed in this function. The module address is set by DIP switches on the rack backplane. A table of these switch settings is given in below. Notice that a byte address is used. 1000 (decimal) is not added (as it is for the DPREQ instruction).

Switch Settings for the Window Address Instruction

Window Address In (Hex)	I/O Point	Dip Switch Position 7	6	5	4	3	2	1
00	1-8							
01	9-16							x
02	17-24						x	
03	25-32						x	x
04	33-40					x		
05	41-48					x		x
06	49-56					x	x	
07	57-64					x	x	x
08	65-72				x			
09	73-80				x			x
0A	81-88				x		x	
0B	89-96				x		x	x
0C	97-104				x	x		
0D	105-112				x	x		x
0E	113-120				x	x	x	
0F	121-128				x	x	x	x
10	129-136			x				
11	137-144			x				x
12	145-152			x			x	
13	153-160			x			x	x
14	161-168			x		x		
15	169-176			x		x		x
16	177-184			x		x	x	
17	185-192			x		x	x	x
18	193-200			x	x			
19	201-208			x	x			x
1A	209-216			x	x		x	
1B	217-224			x	x		x	x
1C	225-232			x	x	x		
1D	233-240			x	x	x		x
1E	241-248			x	x	x	x	
1F	249-256			x	x	x	x	x

Window Address In (Hex)	I/O Point	Dip Switch Position 7	6	5	4	3	2	1
20	257-264		x					
21	265-272		x					x
22	273-280		x				x	
23	281-288		x				x	x
24	289-296		x			x		
25	297-304		x			x		x
26	305-312		x			x	x	
27	313-320		x			x	x	x
28	321-328		x		x			
29	329-336		x		x			x
2A	337-344		x		x		x	
2B	345-352		x		x		x	x
2C	353-360		x		x	x		
2D	361-368		x		x	x		x
2E	369-376		x		x	x	x	
2F	377-384		x		x	x	x	x
30	385-392		x	x				
31	393-400		x	x				x
32	401-408		x	x			x	
33	409-416		x	x			x	x
34	417-424		x	x		x		
35	425-432		x	x		x		x
36	433-440		x	x		x	x	
37	441-448		x	x		x	x	x
38	449-456		x	x	x			
39	457-464		x	x	x			x
3A	465-472		x	x	x		x	
3B	473-480		x	x	x		x	x
3C	481-488		x	x	x	x		
3D	489-496		x	x	x	x		x
3F	505-512		x	x	x	x	x	x
40	513-520	x						

GEK-25379

Window Address		Dip Switch Position						
In (Hex)	I/O Point	7	6	5	4	3	2	1
41	521-528	x						x
42	529-536	x					x	
43	537-544	x					x	x
44	545-552	x				x		
45	553-560	x				x		x
46	561-568	x				x	x	
47	569-576	x				x	x	x
48	577-584	x			x			
49	585-592	x			x			x
4A	593-600	x			x		x	
4B	601-608	x			x		x	x
4C	609-616	x			x	x		
4D	617-624	x			x	x		x
4E	625-632	x			x	x	x	
4F	633-640	x			x	x	x	x
50	641-648	x		x				
51	649-656	x		x				x
52	657-664	x		x			x	
53	665-672	x		x			x	x
54	673-680	x		x		x		
55	681-688	x		x		x		x
56	689-696	x		x		x	x	
57	697-704	x		x		x	x	x
58	705-712	x		x	x			
59	713-720	x		x	x			x
5A	721-728	x		x	x		x	
5B	729-736	x		x	x		x	x
5C	737-744	x		x	x	x		
5D	745-752	x		x	x	x		x
5E	753-760	x		x	x	x	x	

Window Address		Dip Switch Position						
In (Hex)	I/O Point	7	6	5	4	3	2	1
5F	761-768	x		x	x	x	x	x
60	769-776	x	x					
61	777-784	x	x					x
62	785-792	x	x				x	
63	793-800	x	x				x	x
64	801-808	x	x			x		
65	809-816	x	x			x		x
66	817-824	x	x			x	x	
67	825-832	x	x			x	x	x
68	833-840	x	x		x			
69	841-848	x	x		x			x
6A	849-856	x	x		x		x	
6B	857-864	x	x		x		x	x
6C	865-872	x	x		x	x		
6D	873-880	x	x		x	x		x
6E	881-888	x	x		x	x	x	
6F	889-896	x	x		x	x	x	x
70	897-904	x	x	x				
71	905-912	x	x	x				x
72	913-920	x	x	x			x	
73	921-928	x	x	x			x	x
74	929-936	x	x	x		x		
75	937-944	x	x	x		x		x
76	945-952	x	x	x		x	x	
77	953-960	x	x	x		x	x	x
78	961-968	x	x	x	x			
79	969-976	x	x	x	x			x
7A	977-984	x	x	x	x		x	
7B	985-992	x	x	x	x		x	x
7C	993-1000	x	x	x	x	x		

Format of the Command Block Reference

The second (command block) reference used by the Window function must be a register. Power flows out of the Window function only if the window address specified in the first reference is out of range, or if the window fails. Failure may be caused by:

- Addressed device not responding (timeout).
- Addressed device sends bad header (checksum).
- Addressed device fails to close window (timeout).

Entering a Window Function

The Window function can be placed in columns 1 to 6 in a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Control Functions (F6), and Window (F7).

```
                *******          R******
-| WINDOW ADDRESS          COMM BLOCK |
```

3. The cursor is at ADDRESS. Using the decimal keypad, type in number of the reference. It may be any reference type. If you do not want to enter the module address now, press CTRL-E (or the Enter key).

   To enter the address now, convert the work area to hexadecimal format by pressing the Hex key. Enter two hex digits to represent the channel number, and two digits to represent the card address of the module. After entering both the reference and the value, press the Shift and Enter keys.

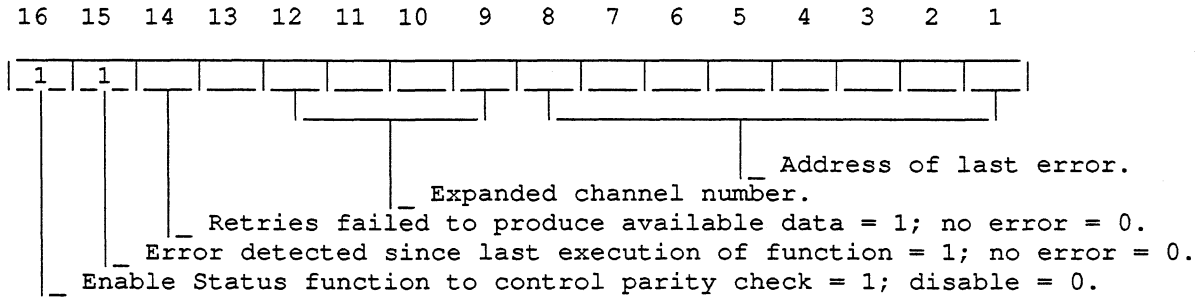
4. The cursor moves to BLOCK. Enter the register that will store the command block or computer mailbox address. It must be an available register. Then, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Status

NOTE

Please refer to the information in chapter 13 on the Status function, which is available in the Advanced function set.

In CPUs with version 110 (4.0) or later Expanded or Expanded II software, the Status command will also report the Expanded I/O channel on which a parity error occurred. If power does not flow to the Status function and bit 16 is set to a 1, then the bits have the following meanings:



In addition, if the CPU has Expanded I/O enabled, then bits 12 - 9 are defined as the Expanded I/O channel on which the parity error was detected. The CPU cannot differentiate between I/O parity on the Main I/O chain and on the Auxiliary I/O chain, so the channel returned will be in the range 0 - 7 for parity on either chain. If the error occurs in broadcast mode, then the pattern 1000 will be returned in bits 12 - 9.

Computer Mailbox

The computer mailbox is an area in register memory that can be set aside for the automatic transfer of data from a bus controller. Use of the computer mailbox must be selected on the Expanded Functions menu. When enabled, the computer mailbox consists of the highest 70 register addresses in memory.

The computer mailbox begins with registers that give the communications parameters. These are followed by 64 additional registers for data storage.

Computer Mailbox

Bus Controller Address	Register 1
Operation (Read/Write)	Register 2
Communications Status	Register 3
I/O Address of Target	Register 4
Mailbox Address for Data	Register 5
Data Length in Bytes	Register 6
data registers Beginning address given in register 5, length in register 6.	to Register 70

The contents of the Computer Mailbox are defined below. Register contents can be entered directly, or by the communicating device, or using ladder logic in the program. The parameters can be:

Register N: The backplane address of the resident bus controller. It is entered using the formula (channel number X 1024 + I/O address). Use the decimal equivalent of the channel number. For example, bus controller location 257 on channel E would be calculated like this:

```
(14x1024)+257 =
(14336)  +257 = 14593
```

Register N+1: The command number. This number determines which of the following operations is performed when the function receives power flow:

Number	Description
1	Idle (no operation performed).
2	Read configuration.
3	Write configuration.
4	Read diagnostics.
5	Read memory.
6	Write memory.
7	Read analog status (all analog inputs from a block into a CPU register).
8	Test.
9	Read status table reference.
10	Write status table reference.
11	Switch BSM.
12	Send datagram.
13	Receive datagram.

Register N+2: Status code. This register should first be cleared to zero by the CPU. It will be loaded by the bus controller at the end of each scan when a status is available. The content of this register can be:

Number	Description
0	Request from CPU to bus controller.
1	Bus controller accept / in progress.
2	Completed without error.
12	Command terminated due to a syntax error.
20	Command terminated due to data transfer error.

Register N+3: For commands 1-7, this register contains the I/O block or bus controller status table address. For commands 8-13, this register contains the device number (serial bus address) of the I/O block or bus controller.

Register N+4: This register points to an address in CPU memory. For commands 2, 4, 7, and 9, this address is where data received following Read commands will be placed in the CPU. For commands 3 and 12, this register contains the address where the data to be sent to the Genius I/O subsystem begins in the CPU. For command 13, this register contains the address where the information to be read (header) resides. The actual data will be received at the address pointed to be register 8.

Register N+5: For commands 5, 6, 8, 12, 13. The amount of data transferred, in bytes.

Register N+6: For commands 5 and 6, the offset to memory LSB. For commands 12 and 13 only, the command code of the datagram message.

Register N+7: For commands 5 and 6, the offset to memory MSB. For command 13 only, the command code for the first register in the CPU where the message received in reply to the datagram will be stored.

Register N+8: For command 13 only, the message received length. Written by the bus controller.

Register N+9: For command 13 only, the remaining part of the command code specified in register 7. Written by the CPU.

For more information, refer to the *Genius I/O System User's Manual* (GEK-90486).

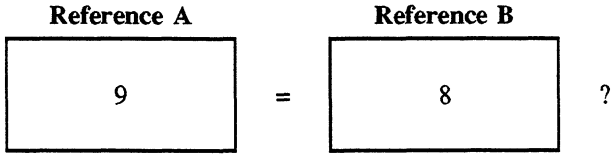
SECTION 3
Expanded Arithmetic Function

The Expanded function set includes an enhanced version of the Compare function (one of the Basic functions).

Expanded Compare

The Expanded Register Compare function performs a signed 16-bit comparison between two values. The addressing range includes all I/O, Auxiliary and Expanded I/O, constants, and registers. For register-register comparisons below 1K, the Basic Compare function may be used instead of the Expanded Compare function.

Every scan that power flows to the Expanded Compare function, the program compares the content of reference A to the content of reference B. Power flows only if the two values are equal.



Entering an Expanded Compare Function

An Expanded Compare function can be placed in columns 1 to 8 of the first line of a rung.

1. Enter any logic required to control power flow to the function. If the function is placed at the left rail, it will execute unconditionally upon every sweep.
2. Select Advanced Mnemonic Group (F7), Expanded Arithmetic (F2), and Expanded Compare (F7).

```
******* *******
-|  A EQUAL B  |-
```

3. The cursor is at A. Using the numeric keypad, type in the address of the first value. It may be any available I/O, Auxiliary I/O, or Expanded I/O point on a byte boundary, any constant or register. After entering the reference, press CTRL-E (or the Enter key).
4. The cursor moves to B. Using the same parameters, enter the address of the value to be compared to the first value. After entering the reference, press CTRL-E (or the Enter key).
5. Complete the logic for the rung, and press CTRL-A (or the Accept key). The Edit key functions reappear at the bottom of the screen.

Appendix A
Setup Information

This appendix summarizes the setup information required in order to use Logicmaster 6 software.

- Configuring the communication port.
- Connecting to a Workmaster or Cimstar I industrial computer.
- Connecting to a Workmaster II industrial computer.
- Connecting to an IBM personal computer.
- Using modems.
- Installing optional boards:
  - Expanded Memory (RAM Disk) card.
  - Enhanced Graphics Adapter card.
  - Diskette Drive Adapter card.
  - Color Graphics/Monitor Adapter card.
- Printer protocol and cable diagrams.

Setup Information for the Parallel Version of Software

For parallel communications, the cable connecting the Workmaster computer to the Series Six PLC is a standard I/O cable. Any of the following cables can be used:

Order Number	Description
IC600WD002A	2 ft (0.6 M) cable with connectors
IC600WD005A	5 ft (1.5 M) cable with connectors
IC600WD010A	10 ft (3.0 M) cable with connectors
IC600WD025A	25 ft (7.5 M) cable with connectors
IC600WD050A	50 ft (15.0 M) cable with connectors
IC600WD100A*	100 ft (30.0 M) cable with connectors
IC600WD200A*	200 ft (60.0 M) cable with connectors
IC600WD500A*	500 ft (150.0 M) cable with connectors

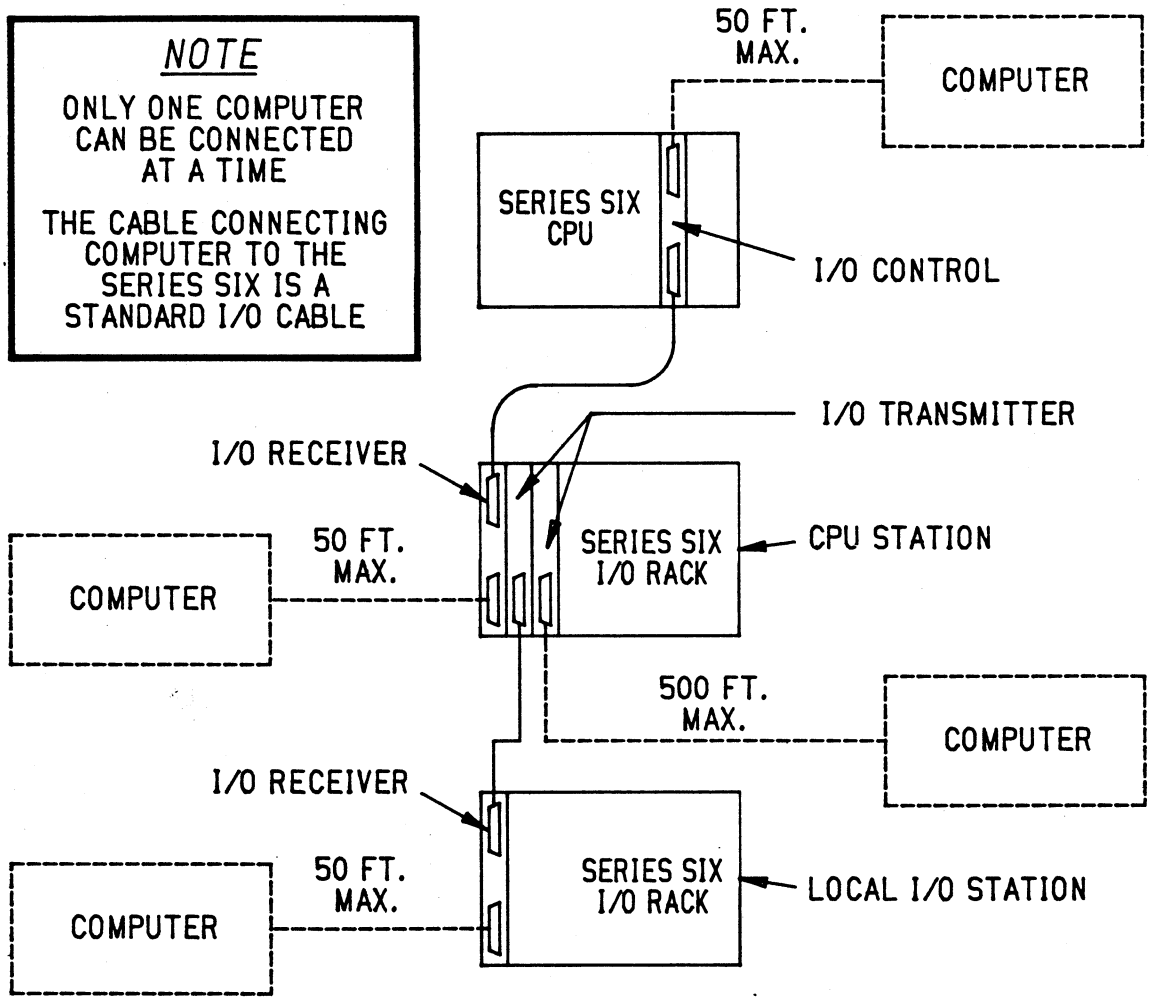
* Only for connection to the Series Six PLC I/O Transmitter card.

To communicate with the Series Six PLC using the parallel version of Logicmaster 6, the computer may be connected in one of these ways:

1. Directly to the Series Six PLC I/O Control Module (up to 50 feet).
2. Through the I/O Receiver Module in an I/O rack residing in a CPU station (up to 50 feet).
3. Through the I/O Receiver Module in an I/O rack residing in a Local I/O station (50 feet maximum).
4. Through an I/O Transmitter Module in an I/O rack residing in either a CPU station or Local I/O station (up to 500 feet). Increasing the distance between the computer and the CPU causes propagation delays, which slow communications somewhat.

GEK-25379

a40065



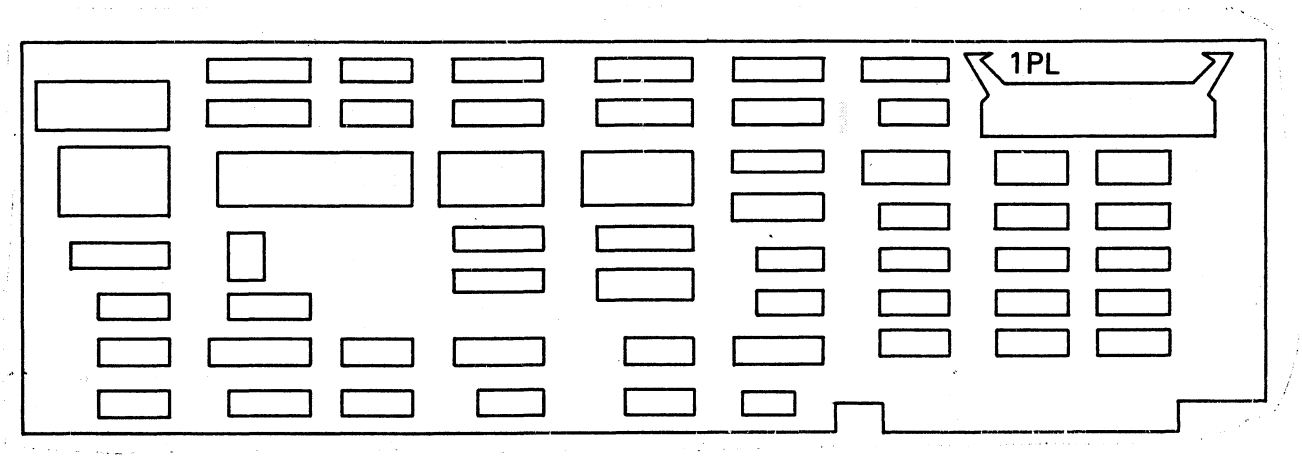
Workmaster/Series Six PLC Interface/Terminator Boards

The Workmaster/Series Six PLC Interface and Terminator boards, connected by a 34-wire ribbon cable, provide high-speed parallel communications between the Series Six PLC and the computer.

The board set must be installed in adjacent slots of the computer. If necessary, move other boards already in the computer so that two adjacent slots are empty. If the boards are being installed in an IBM PC-AT computer, they require an XT slot (one connector) and adjacent AT slot (two connectors).

1. First, install the Interface board. If the board is being installed in an IBM PC-AT computer, place it in an XT-type (one connector) slot.
2. After verifying that the DIP switches on the Terminator board are correctly set, as described on the next page, install it next to the Interface board. In an IBM PC-AT, this must be an AT-type (two connector) slot.
3. Plug the 34-pin ribbon cable connector from the Terminator board into the 34-pin connector on the top of the Interface board.

a41022



The Workmaster/Series Six PLC Terminator board (IC640BLD304) provides the circuitry to terminate and protect data lines between the Series Six PLC and the computer.

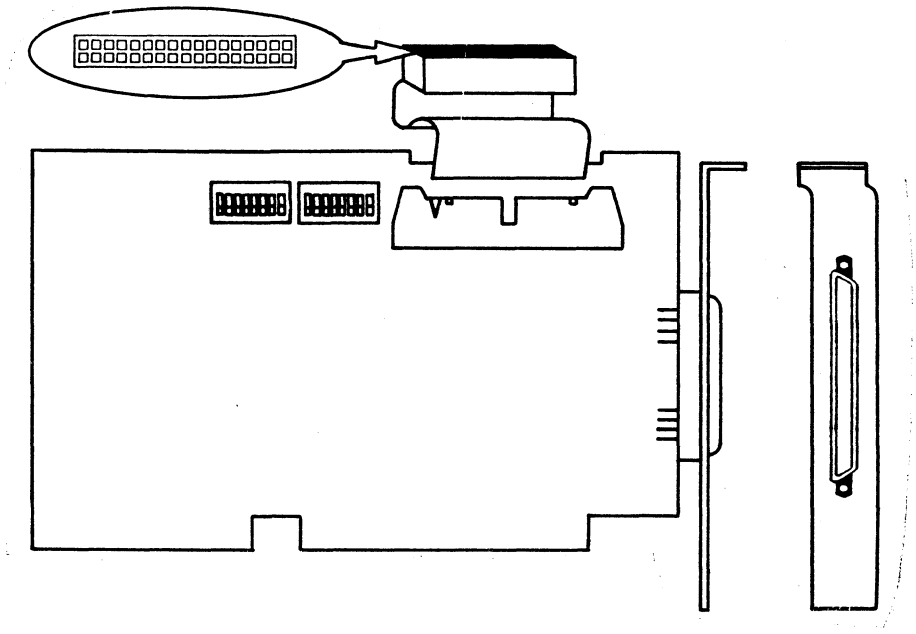
The DIP (rocker) switches on the Terminator board are factory-set in the OFF position. With the switches in this position, the computer can be connected to the I/O Control board, or any Series Six PLC Parallel Receiver board up to 10 feet (3 meters) away.

Parallel Transmitter Board

The computer can also be directly connected to a Series Six PLC Parallel Transmitter board up to 500 feet (150 meters) away. The Parallel Transmitter board (IC600BF900) must be dedicated to communication with the computer.

For connection to the Parallel Transmitter board, the DIP switches on the Workmaster/Series Six PLC Terminator board to the ON (closed) position.

a41021



WARNING

All DIP switches on the Workmaster/Series Six PLC Terminator board must be properly set, as described above. Failure to do so may result in the CPU stopping in an incorrect state, or may cause incorrect operation of the I/O bus, with outputs directed to an incorrect state.

Setup Information for the Serial Version of Software

For serial communications, the computer must be connected to the CCM2 (Communications Control Module 2) card in the Series Six PLC CPU. Or, it may be connected to a CCM3 card (version CCM3IC600CB517C or later) that is operating in CCM2 mode.

The CCM2 card provides either RS-232 or RS-422 communications. RS-232 is used for direct connections up to 50 feet (15 meters). RS-422 is used for direct connections up to 4000 feet. RS-232 with modems may be used where longer transmission distances are required.

For the Logicmaster 6 system to communicate with the CCM2 card, the port characteristics must match, as described in this appendix.

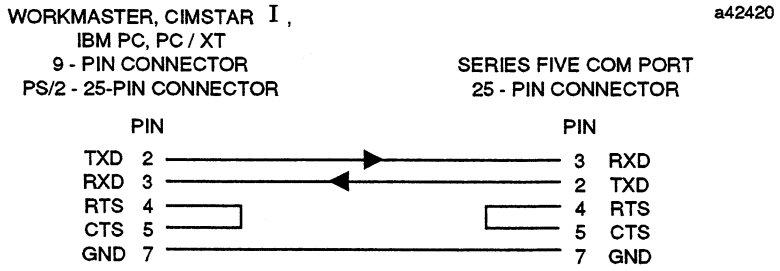
The CCM2 card is connected by cable to port 1 (COM1) or port 2 (COM2) on the computer. For an IBM PC-XT computer, follow the instructions provided with the computer to configure the port you want to use for communications.

Logicmaster 6 serial communications use interrupt-driven I/O. When you configure the serial communications card, be sure to use the correct interrupt request line for the selected port. The Logicmaster 6 software uses these IBM-standard request line assignments: port COM1 (I/O addresses 3F8-3FF) uses interrupt request IRQ4, and port COM 2 (I/O addresses 2F8-2FF) uses interrupt request IRQ3. On serial adapter cards from IBM, the interrupt request line and port address selections are coupled together to enforce these assignments. However, on cards from other vendors (such as Tecmar or AST), the port address and the interrupt request line may be assigned independently. Make sure that you use IRQ4 for COM1 and IRQ3 for COM2.

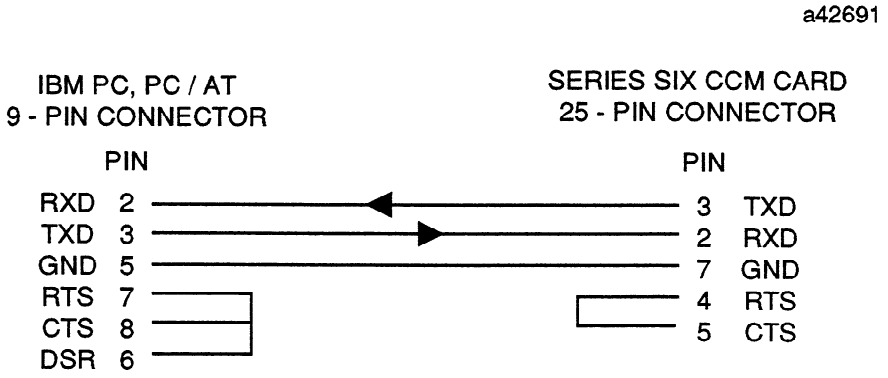
RS-232 Communications

For point-to-point communications to a single Series Six PLC, use the RS-232 line interface standard and either of the CCM protocols (peer-to-peer or master/slave). Peer-to-peer is somewhat more efficient. RS-232 is especially convenient since most computers have at least one RS-232 port which may be used by the Logicmaster 6 software.

For direct RS-232 communications (not using a modem), use one of the following cables. For the IBM Personal System 2 computer, use the COM1 port (25-pin connector). Pin assignments are the same.



The IBM PC-AT reverses the order of the transmit (TXD) and receive (RXD) pins, and uses pin 5 for signal ground.



RS-422 Connections

For serial communications in a direct multidrop configuration (not using modems) to more than one Series Six PLC CPU, you must use the RS-422 line interface standard and the CCM master/slave protocol. RS-422 cabling requirements for the CCM card are documented in the *Series Six PLC Data Communications Manual* (GEK-25364).

When RS-422 connection is made from the CCM card to a device not supplied by GE Fanuc, cross reference the RS-422 signal nomenclature, as shown below:

CCM Signal Name	RS-422 Standard EIA Signal Name
RS-422 out + (TXD+)	B
RS-422 out - (TXD-)	A
RS-422 in + (RXD+)	B'
RS-422 in - (RXD-)	A'

During a mark condition (logic 1), B will be positive with respect to A. During a space condition (logic 0), B will be negative with respect to A.

An RS-422 link must terminate with the proper resistance to minimize reflection on the line. Some devices (such as the Asynchronous/Joystick card) already have terminating resistors in the circuit. If you are making connection to a device that does not have such built-in resistance, you should provide it. A resistor can be installed in the connector at either end of a point-to-point or multidrop link, between the Receive Data (+) and Receive Data (-) pins. No terminating resistor is needed for intermediate drops on a multidrop link.

Configuring the CCM2 Card

The CCM2 card has two communications ports. The upper port (J1) is a 25-pin male connector. The lower port (J2) is a 9-pin male connector. The communications characteristics of these ports can be set up either in hardware or in software. Both methods are described in this section. Software configuration also requires selection in the hardware. Once this selection has been made, communications parameters are set up in the software.

Baud Rate: Data transfer rate, in bits per second. The factory setting for the CCM2 card is 19.2K bps.

Protocol: Protocol may be either master/slave or peer-to-peer.

Peer-to-Peer: In this mode, only one CPU may be connected serially to the computer.

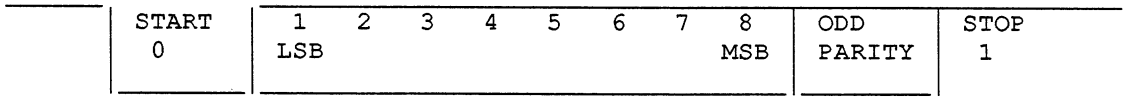
Master/slave: In this mode, up to 8 CPUs may be connected serially to one computer.

RS-232: This may be used with either peer-to-peer or master/slave protocol.

RS-422: This may be used with either peer-to-peer or master/slave protocol. It is used primarily for direct connection up to 4000 feet (1200 meters).

Turn-Around Delay: This refers to the amount of time before sending a control character, start of header, or start of a data block. For direct connections, 0ms is usually selected. 10ms may be selected for situations characterized by slow response times. For more information, refer to the *Series Six Data Communications Manual* (GEK-25364).

Parity: The CCM2 card uses the serial data format shown below:



<--------- Direction of data flow.

The data is divided into 8-bit bytes and transferred using an asynchronous format. This format consists of one start bit, 8 data bits, one parity bit, and one stop bit. The parity bit can be enabled or disabled. When enabled, the format is odd parity. When disabled, the parity bit is not transmitted.

Hardware Configuration for the CCM2 Card

If the CCM2 card is configured in hardware, jumpers and DIP switches on the CCM2 card are used to select the baud rate, protocol, delay, and parity for ports J1 and J2. The switch numbers without parentheses below are the numbers printed on the CCM card. The switch numbers shown in parentheses are the numbers on the switch package. They are included to help you configure the module.

Configuration for Port J1:

In the following configuration, O = open and C = closed.

		Switches							
		9 (1)	10 (2)	11 (3)	12 (4)	13 (5)	14 (6)	15 (7)	16 (8)
Baud Rate	300	O	O	O					
	600	C	O	O					
	1200	O	C	O					
	2400	C	C	O					
	4800	O	O	C					
	9600	C	O	C					
	19200	O	C	C					
	38400	C	C	C					
Protocol	Slave RS-232				O	C	O		
	Slave RS-422				C	C	O		
	Peer RS-232				O	O	C		
	Peer RS-422				C	O	C		
	Peer RS-422 with CLK				O	C	C		
	Software Configure Mode				C	C	C		
Turn Around Delay	0ms at full duplex							O	O
	10ms at half duplex							C	O
	500ms at half duplex							O	C
	500ms with timeouts disabled							C	C
Parity Selection	Parity is always ODD when using hardware configuration; the serial port on the computer should be set for ODD parity at all times unless Software Configuration mode is being used. Refer to the *Data Communications Manual* (GEK-25364) for more information.								

Configuration for Port J2:

In the following configuration, O = open and C = closed.

		Switches								
		1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	6 (6)	7 (7)	8 (8)	17
Baud Rate	300	O	O	O						
	600	C	O	O						
	1200	O	C	O						
	2400	C	C	O						
	4800	O	O	C						
	9600	C	O	C						
	19200	O	C	C						
	38400	C	C	C						
Protocol Slave	RS-232				O	C	O			
	Slave RS-422				C	C	O			
	Peer RS-232				O	O	C			
	Peer RS-422				C	O	C			
	Peer RS-422 with CLK				O	C	C			
	Software Configure Mode				C	C	C			
Turn Around Delay	0ms at full duplex							O	O	
	10ms at half duplex							C	O	
	500ms at half duplex							O	C	
	500ms with timeouts disabled							C	C	
Parity Selection	Odd									C
	None									O

Software Configuration for the CCM2 Card

The CCM card can be configured using CPU registers R0247 and R0248 in CCM Software Configuration mode. To enter this mode, first position the module switches 12, 13, 14, and 17 as shown below:

Function	Switches			
	12	13	14	17
Software Configure Odd Parity	C	C	C	C

In addition, the other switches and jumpers on the CCM card must be correctly set, as shown in the *Data Communications Manual*, GEK-25364.

In the Software Configuration mode, register R0247 represents the configuration for the serial port J1, and register R0248 represents the configuration for serial port J2. The format of the configuration data as shown below is exactly the same for both registers.

bit	16	15 14 13 12 11 10	9	8 7	6 5 4	3 2 1
	Port Enable/Disable	for the Operator Interface Unit Module	Parity	Turn Around Delay	Protocol	Baud Rate

If the CCM2 card is idle and not executing a SCREQ command or serial communications, the CCM2 will reinitialize once per second. When using software configuration, the reinitialization routine in the CCM2 card will read the configuration data from the CPU registers to configure the serial ports.

To reconfigure the CCM2 on-line, the ladder logic must change the configuration registers, and then ensure that the CCM2 is idle for a minimum of 3 seconds.

The table below shows the bit patterns required for selecting module options.

		BITS				*	*									
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Baud Rate	300														0	0	0
	600														0	0	0
	1200														0	1	0
	2400														0	1	1
	4800														1	0	0
	9600														1	0	1
	19200														1	1	0
	38400														1	1	1
Protocol	Slave RS-232											0	1	0			
	Slave RS-422											0	1	1			
	Peer RS-232											1	0	0			
	Peer RS-422											1	0	1			
	Peer RS-422 with CLK (J1 only, R247)											1	1	0			
	Test 1 (J2 only, R248)											1	1	0			
Turn Around Delay	0ms at full duplex									0	0						
	10ms at half duplex									0	1						
	500ms at half duplex									1	0						
	500ms with timeouts disabled									1	1						
Parity Selection	Odd								1								
	Even								0								
OIU (Operator Interface Unit)**																	
	OIU							0									
	Dumb Terminal							1									
OIU Enable	Enable				0												
	Disable				1												
OIU Polling	Non-polled			0													
	Polled			1													
OIU Memory Protect																	
	Enable		0														
	Disable		1														
Port Enable	Enable	0															
	Disable	1															

* Bits 11 & 12 are not used.

** When OIU or Dumb Terminal mode is selected, the CCM2 operates in an even parity format. If a dumb terminal is used, it must be configured as a 7-bit even-parity device.

Connection to the Cimstar I Computer

For the Cimstar I computer, the 9-pin upper connector on the top of the Multifunction module is used for RS-232 direct communications over distances of less than 50 feet. For communications over distances greater than 50 feet, the RS-422 port on the Industrial Option board is used instead. The connector for this port is located on the underside of the Multifunction module if the industrial option is installed.

The default factory settings are COM1 for the RS-232 port and COM2 for the RS-422 port. These designations may be changed, as described in GEK-90527, *Cimstar I Reference Manual.*

Cabling and pinouts for the Cimstar I computer are the same as those shown for the Workmaster computer in this appendix.

Multifunction Module

The standard Multifunction module provides two ports.

1. The top area of the module has a 25-pin port that is a Centronics-type parallel interface normally used for a printer. The parallel port in the Multifunction module is shipped as LPT1.
2. There is also a 9-pin port that can be used for direct communication, when the distance between the Cimstar I computer and the Series Six CPU is less than 50 feet. The serial port on the Multifunction module is shipped as COM1.

Industrial Option Board

The Industrial Option board to the Multifunction module provides an RS-422 serial port, an AC power-fail circuit, and a joystick interface. The connector for the RS-422 serial port is located on the bottom of the Multifunction module. The port is normally configured as COM2, but can be changed. The RS-422 port may be totally disabled in order to have a second RS-232 port on a user-supplied board designated as COM2.

Connection to the Workmaster Computer

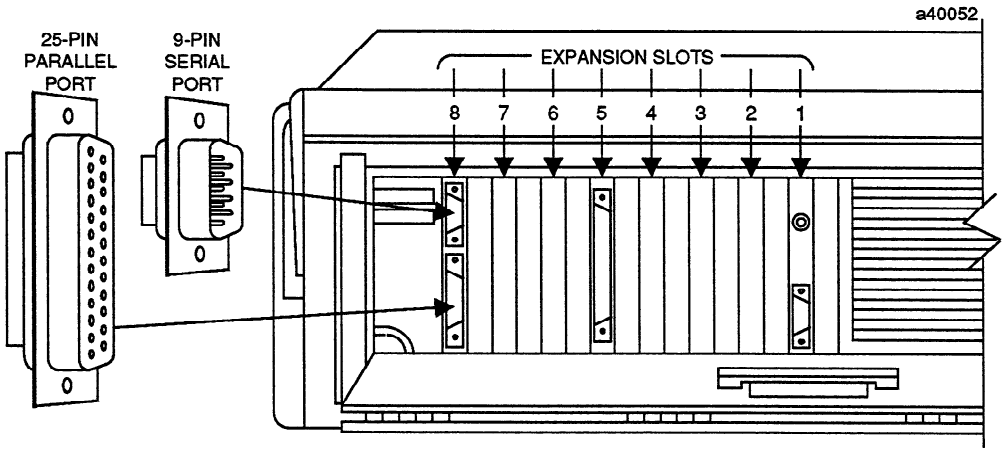
For the Workmaster industrial computer, the 9-pin upper connector on the Combination Adapter card is used for RS-232 direct communications over distances of less than 50 feet. For communication over distances greater than 50 feet, the Asynchronous/Joystick card must be used.

The designation as port COM1 is usually assigned to the RS-232 port on the Combination Adapter card, as described in this appendix. You can change this by assigning port COM1 to the RS-232/RS-422 port on the Asynchronous/Joystick card instead. This reassigns port COM2 to the Combination Adapter card.

In addition, you can select either port 1 or port 2 for communication on the Communication Setup menu.

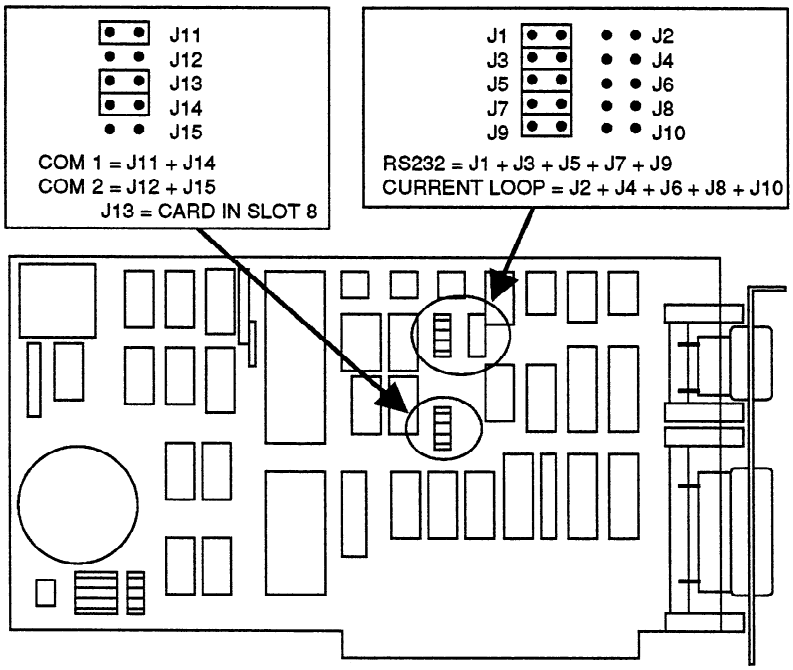
Combination Adapter Card

The Combination Adapter card (expansion slot 8) contains 2 ports, as shown in the following illustration:



The 9-pin upper port on the Combination Adapter card is normally used for direct communication, when the distance between the Workmaster computer and the Series Six CPU is up to 50 feet. The Combination Adapter card is shipped to operate as COM1, as shown below.

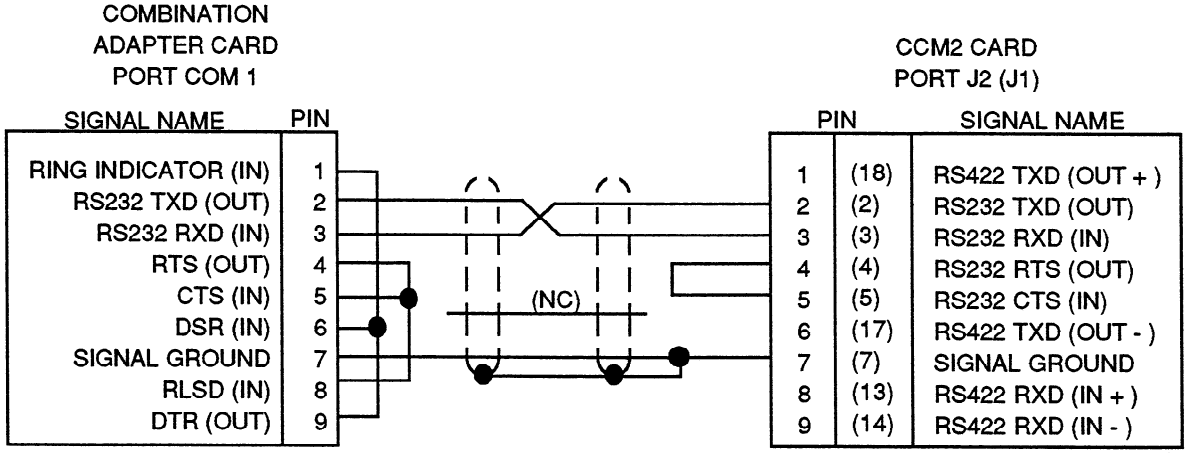
a41043



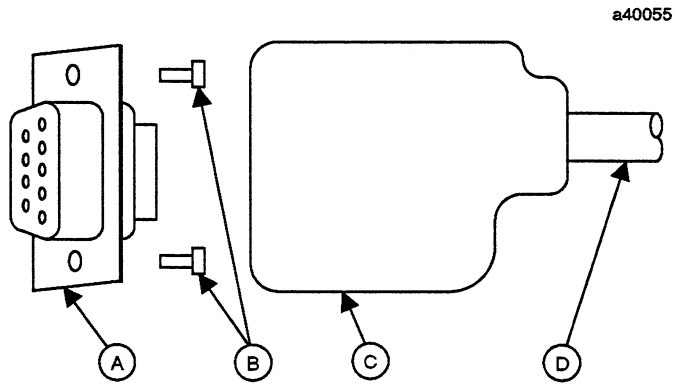
The 25-pin lower port, designated LPT1, is a parallel Centronics interface normally used for a printer. This port is described later in this appendix.

The following illustration shows the cable assembly for RS-232 communication between the 9-pin female connector on the Combination Adapter card and either port J1 (the 9-pin male connector) or port J2 (the 25-pin male connector) on the CCM2 card. Pin numbers for J2 are shown in parentheses.

a41023



The following illustration shows the connector and cable for attachment to the Combination Adapter card.

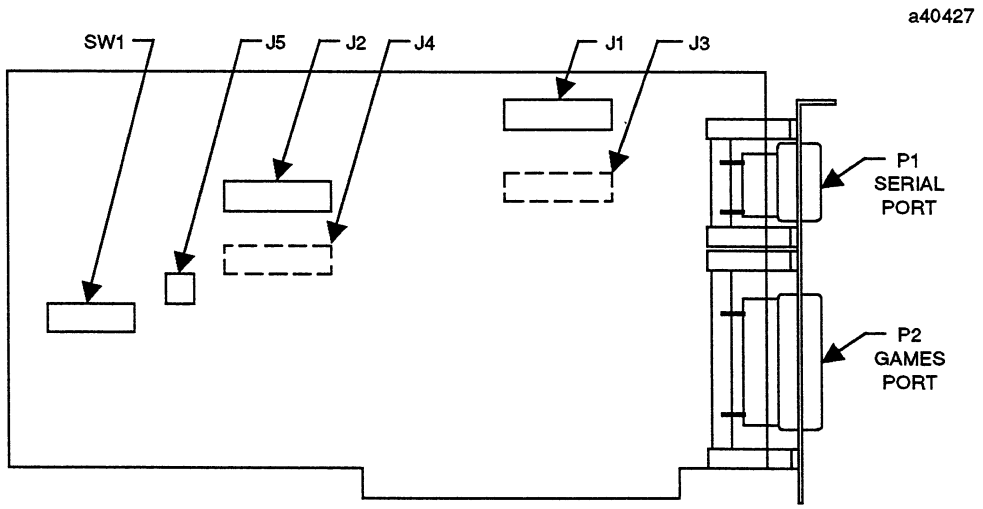


A CONNECTOR: CANNON DE9S (FEMALE)
B SCREWLOCK: CANNON D20419
C HOOD: CANNON DE110963-1
D CABLE: BELDEN 9300 SERIES (SHIELDED TWISTED PAIR)

Asynchronous/Joystick Card

The Asynchronous/Joystick card, IC640BGB311, is a user-installed option card, which provides RS-232 or RS-422 communications to the Series Six CPU, or to another serial device.

This card has two ports. The 9-pin upper port is used for connection to the Series Six CPU. The lower port is for use with the joystick. The following illustration shows the location of the ports and the configuration jumpers and switches.



There are two types of Asynchronous/Joystick card, those with jumper J5 as show below (type B), and those without (type A).

Version A of the Asynchronous/Joystick Card

The electrical interface for the RS-232/RS-422 port is set up by placing two DIP shunt packages (one 16-pin pack and one 14-pin pack) in the appropriate sockets. The 14-pin pack is placed in socket J2 or J4, and the 20-pin pack is placed in socket J1 or J3.

Interface Type	Shunt Locations	
	20-Pin	14-Pin
RS-232	J3	J4
RS-422	J1	J2

The other card options for IC640BGB311A are selected using the DIP switch package SW1.

Switch Number	Open Function	Closed Function
1	Game port enabled.	Game port disabled.
2	Serial port disabled.	Serial port enabled.
3	COM2 selected.	COM1 selected.
4	Enable COM2.	Enable COM1.
5	Enable COM1.	Enable COM2.
6	Force CTS true.	CTS from interface.
7	Force DSR true.	DSR from interface.
8	Force RLSD true.	RLSD from interface.

NOTE

Switches 4 and 5 are for RS-422 only and should never both be on at the same time.

Switches 6, 7, and 8 pertain to RS-232 only. In the RS-422 configuration, CTS, DSR, and RLSD are forced to a TRUE state, regardless of the positions of these switches.

Version B of the Asynchronous/Joystick Card

This version of the Asynchronous/Joystick card includes a jumper (J5) which is not on the A version. The electrical insterface for the RS-232/RS-422 port is set up by placing two DIP-shunt packages in the appropriate sockets, as shown below. The 14-pin pack is placed in socket J2 or J4; the 20-pin pack is placed in socket J1 or J3. (Refer to the preceding illustration for the locations of J1, J2, J3, and J4.)

Interface Type	Shunt Locations	
	20-Pin	14-Pin
RS-232	J3	J4
RS-422	J1	J2

The other card options for IC640BGB311B are selected using the DIP switch package SW1.

Switch Number	Open Function	Closed Function
1	Game port enabled.	Game port disabled.
2	Serial port disabled.	Serial port enabled.
3	COM2 selected.	COM1 selected.
4	Transmit data not enabled by RTS.	Transmit data enabled by RTS.
5	Transmit data not enabled by GND.	Transmit data enabled by GND.
6	Force CTS true.	CTS from interface.
7	Force DSR true.	DSR from interface.
8	Force RLSD true.	RLSD from interface.

NOTE

Switches 4 and 5 are for RS-422 only and should never both be on at the same time.

Switches 6, 7, and 8 pertain to RS-232 only. In the RS-422 configuration, CTS, DSR, and RLSD are forced to a TRUE state, regardless of the positions of these switches.

Function	Jumper 5 on Pins
Select COM1	2 - 3
Select COM2	1 - 4

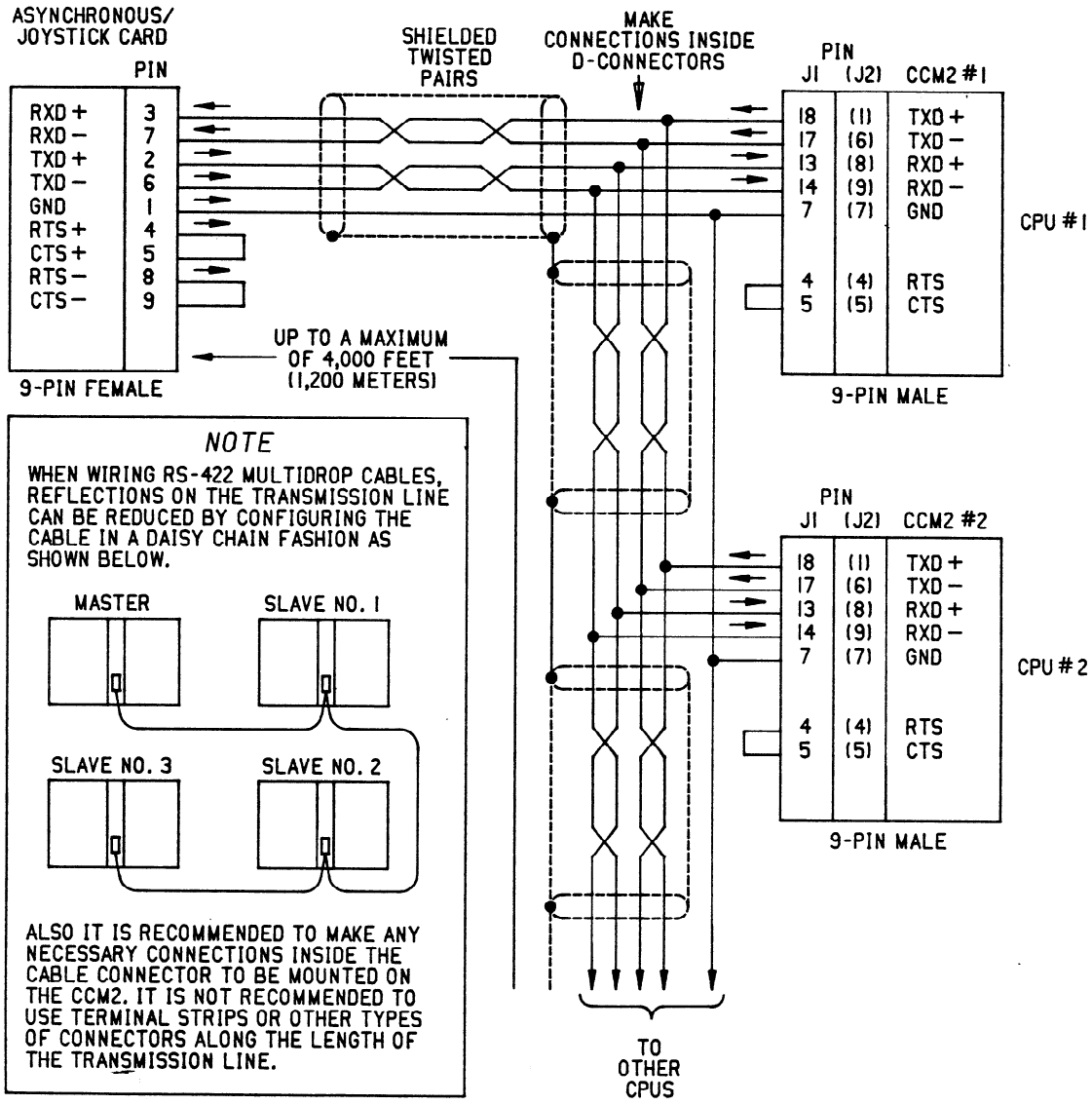
Table D-1. Asynchronous/Joystick Card Versions A or B Switch 1 Settings (RS-232 or RS-422)

	Switch							
	1	2	3	4	5	6	7	8
COM1	Off	On	On	On	Off	On	On	On
COM2	Off	On	Off	Off	On	On	On	On

GEK-25379

The following illustration shows the pin-outs for an RS-422 cable between the Workmaster computer and two CPUs. Refer to the description of the Combination Adapter card for connector and cable part numbers.

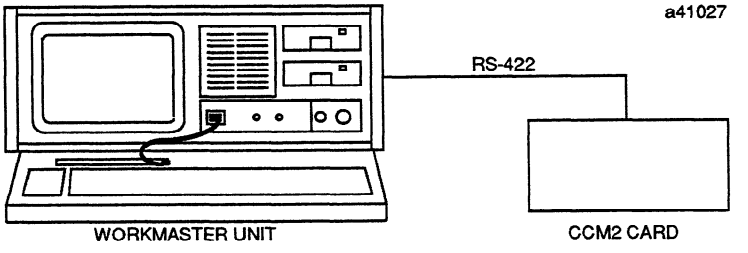
a41028

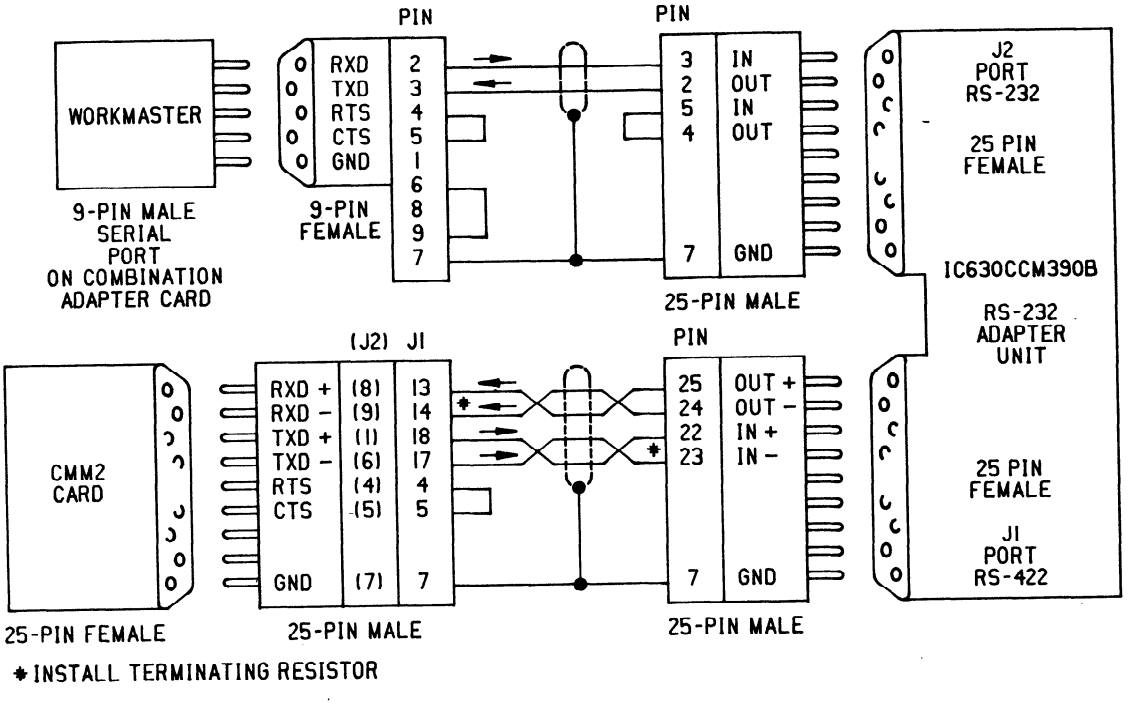


An RS-422 link must terminate with the proper resistance to minimize reflection on the line. Some devices, such as the Asynchronous/Joystick card, already have terminating resistors in the circuit. If you are making connection to a device that does not have such built-in resistance, you should provide it. A resistor can be installed in the connector at either end of a link, between the Receive Data (+) and Receive Data (-) pins. No terminating resistor is needed for intermediate drops. If a Series Six CPU is the end of an RS-422 link, then 150-ohm terminating resistors should be installed across each signal pair (+,-). The resistors can be wired into the connector.

Connecting the Workmaster Computer with an Adapter Unit

The following illustration shows the connection from the serial port (RS-232) on the Combination Adapter card through an RS-232 to RS-422 adapter unit (IC630CCM390B) to the CCM port.





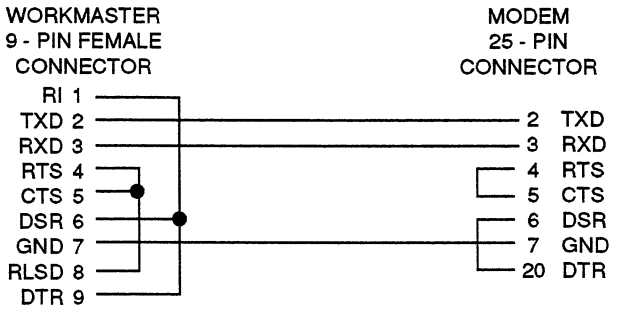
Modems

The Logicmaster 6 software supports full-duplex modems which are compatible with the Bell System 212 standard. In order to use modems with Logicmaster 6 software, the following hardware is required:

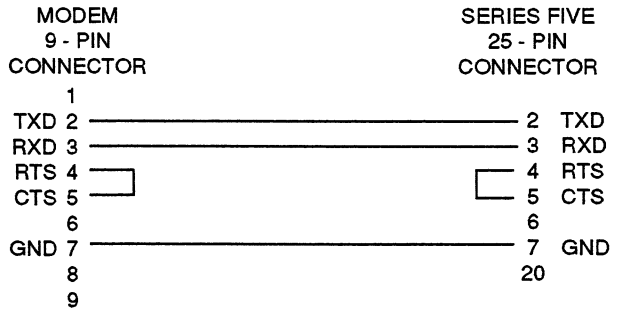
- A Workmaster or Cimstar I industrial computer, or IBM personal computer.
- Two smart modems (Bell System 212 compatible).
- A Series Six PLC system.
- A cable from the host computer to the modem.
- A cable from the modem to the Series Six PLC.
- Two telephone cables.

The next two illustrations show the pin connections between a Workmaster computer and modem, and between a modem and the Series Six PLC.

a42658



a42659



Using a Smart Modem

A "smart" modem is one that can dial and answer the phone. To dial, the modem requires several control characters followed by the ASCII digits of the telephone number. Logicmaster 6 software does not provide a method of directly sending these characters to a serial port.

To use a smart modem:

1. Using an editor, create a text file containing the control characters and the telephone number. In DOS, this may be done by typing:

```
COPY CON:PHONE.TXT
ATD 9785000
F6
```

   You must press the Return key after entering each line. In this example, 9785000 is an example phone number; "F6" is function key 6.

NOTE

The control and dialing commands may vary between modems. Consult your modem user's manual for more information.

2. Set up the modem.
   A. Boot up the Logicmaster 6 system.
   B. Configure the serial port with these settings:

```
Serial port:   1
Baud rate:     1200 or 300
Stop bits:     1
Parity:        None
Data bits:     8
X-On/X-Off:    N
```

   C. Press the Setup Port (F1) key.
3. Background print the file.
   A. In the Communications Setup menu, de-select port 1 for CPU communication port. To do this, leave the work area blank and press the Select Ports (F3) key.
   B. Go to the Print menu, and print the file to the serial port that is connected to the modem.
4. Establish serial communications.
   A. Return to the Communications Setup menu, and re-select the port for CPU communications.
   B. Turn the keyswitch to on-line to place the computer in On-Line mode.

RAM Disk Card

An Expanded Memory card (RAM Disk) is an optional memory card in your computer. It provides volatile memory that can be used for temporary file storage. The RAM Disk card can be used to store Logicmaster 6 overlay files. It can also be used to store parts of a ladder logic program when the Windowing function is active. Windowing allows programs to have up to 10,000 rungs and 5K nicknames. For more information about windowing, refer to chapter 10, *Expanded Functions Menu.*

NOTE

Expanded Memory Board drivers that require more than 5K of RAM for both code and data are not supported by Logicmaster 6 software.

Using a Workmaster or Cimstar I Computer with a RAM Disk Card

If your Workmaster computer or Cimstar I computer came equipped with an Expanded Memory card from GE Fanuc - NA, it is already set up to use the RAM Disk feature.

If your computer was not originally supplied with an Expanded Memory card, the card can be purchased separately from GE Fanuc - NA. In the Workmaster computer, it should replace the 384K RAM card if one is present. Logicmaster 6 software will operate with the card as installed if you are using a floppy-diskette system.

Adding the Expanded Memory Card

With an IBM personal computer, you can add an Expanded Memory card purchased from GE Fanuc - NA, or an equivalent card. The card must be compatible with the Lotus Expanded Memory device interface specifications. If you purchase the card from GE Fanuc - NA, follow the installation instructions provided with the card. The Logicmaster 6 software contains the files required to use the RAM Disk feature.

NOTE

To invoke overlay loading from the RAM Disk, the overlay files must be copied to the RAM Disk prior to invoking Logicmaster 6 software. Refer to chapter 2, *Operation*, for information on copying overlay files to the RAM Disk.

If you plan to use a RAM Disk card purchased from GE Fanuc - NA, the DOS system configuration file (CONFIG.SYS) must be set up to load two special device drivers. These drivers are contained on a diskette shipped with the RAM Disk card. The diskette contains the following four files:

```
GEXMEM.SYS
GEXMEM2.SYS
GEXMDISK.SYS
GEXMTEST.EXE
```

GEXMEM2.SYS and GEXMDISK.SYS are the files required for use with the Logicmaster 6 software. These two files should be transferred to diskette 1, or to the LM6 subdirectory, if you are using a hard disk.

You will need to follow the instructions provided in GEK-96631 in order to configure the RAM Disk card for your system. This publication is shipped with the RAM Disk card. However, it will be helpful to note the following points:

1. A switch bank is located on the top edge of the RAM Disk motherboard, opposite its edge connector. Even though the switches are underneath the daughterboard, you can still flip them with a pointed instrument without removing the daughterboard. To turn a switch on, flip it toward the top edge of the board.
2. If you have only one RAM Disk card in your system, you should only need to adjust switches 7 and 8, which are toward the edge of the motherboard with the metal installation bracket. These switches are used to designate up to one half of the total 1024K bytes of RAM disk memory as normal system memory. This memory will be used by DOS for running application programs. Normally, you should allocate as much memory from the RAM Disk as required to fill your system memory out to 640K bytes. Any remaining memory on the RAM Disk is referred to as "expanded memory" and may be used to emulate a fast-access floppy disk.

For example, if your computer has 256K bytes of memory, you should allocate 384K bytes of the RAM Disk to the system. This provides a total of 640K bytes of system memory with a remainder of 640K bytes of expanded memory. However, if your computer is already loaded with 640K bytes of memory, you should allocate none of the RAM Disk to the system, providing a total of 1024K bytes of expanded memory.

For a Workmaster computer, part of the memory on the RAM Disk card is used as the 640K of RAM memory needed to run Logicmaster 6 software. You should read the instructions provided with the card for information about switch settings, to be sure this memory is properly assigned.

The following table shows the correct settings for switches 7 and 8.

		Motherboard Configuration		Total
SW7	SW8	System Memory	Expanded Memory	Expanded Memory
On	On	none	512K	1024K
Off	On	128K (after 512K)	384K	896K
Off	Off	384K (after 256K)	128K	640K
On	Off	512K (after 64K) (total = 576K)	none	512K

Total Expanded Memory in the above table refers to the amount of memory which may be reserved for use as a fast-access floppy disk. This is done through the CONFIG.SYS command line, which loads the device driver GEXMDISK.SYS.

Once you have installed the RAM Disk in your computer, you can test it by running the program GEXMTEST.EXE, which is provided on the diskette. This is strongly recommended.

Finally, you should edit your CONFIG.SYS file to load the RAM Disk drivers.

```
device=gexmem2.sys -c21c
device=gexmdisk.sys -k640
```

If you are using a hard disk, you should also specify the LM6 path.

```
device= lm6 gexmem2.sys -c21c
device= lm6 gexmdisk.sys -k640
```

The values shown above (-c21c and -k640) specify the RAM Disk's configuration register address (switch 2) and memory allocation (switches 7 and 8). The value (-k640) refers to the amount of expanded memory that you wish to reserve for use as a fast-access floppy disk. You may reserve up to the total listed in the preceding table.

In order to activate the RAM Disk drivers, you must restart the computer after editing the CONFIG.SYS file.

Video-7 Enhanced Graphics Adapter (VEGA) Card

Logicmaster 6 software will support the Video-7 Enhanced Graphics Adapter (VEGA) card operating as the primary display adapter in the computer. The VEGA card must be configured as either a monochrome display adapter or a color graphics adapter. The IBM Enhanced Color Display monitor may be used with the VEGA card configured as a color graphics adapter.

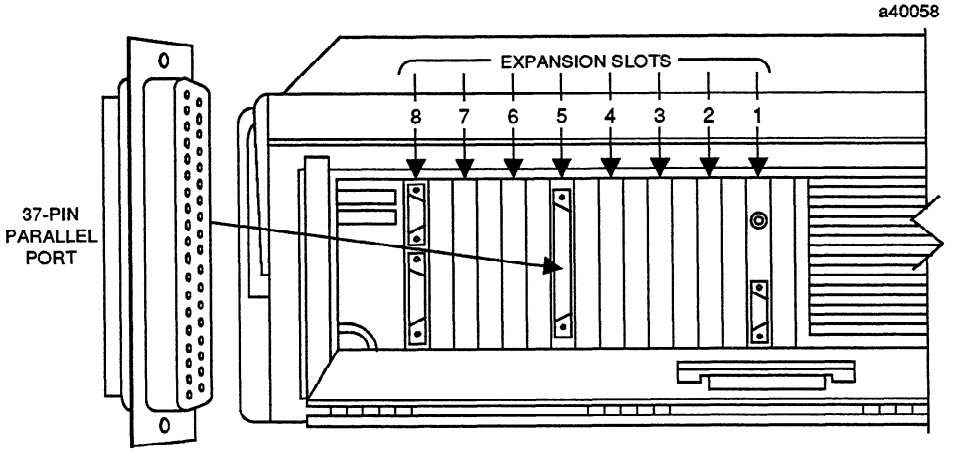
The VEGA toggle switch should be set to the right (as you look at the card edge-on, with the toggle switch at the top). Other switches should be set according to the type of display monitor being used, as follows:

Display Type	Vega Switches					
	1	2	3	4	5	6
Monochrome	off	off	on	off	off	off
Color/Enhanced	off	off	off	on	off	off

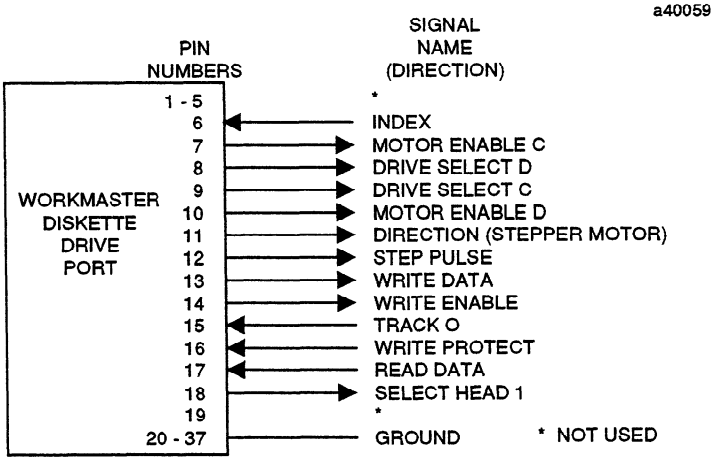
Other switches on the system board of your computer may need to be set as well. For the Workmaster computer, and for IBM PC and PC-XT personal computers, switches 5 and 6 of switch block 1 must be set to ON. For the Cimstar I computer, and for IBM PC-AT personal computers, the diagnostic setup program must be run to configure the machine for the type of monitor (monochrome or color) which will be used.

Diskette Drive Adapter Card

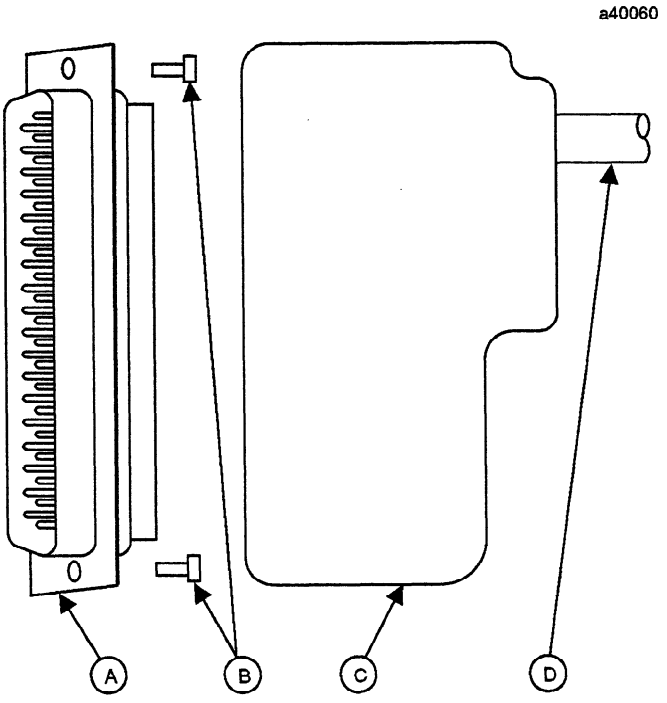
The Diskette Drive Adapter card (expansion slot 5 of the Workmaster computer) provides an interface to an external drive for 5.25-inch diskettes. The location of the 37-pin port is shown in the following illustration.



The following illustration shows the pin-outs for the diskette drive port. All outputs of this port are at standard TTL levels.



The following illustration shows the user connector and cable part numbers:

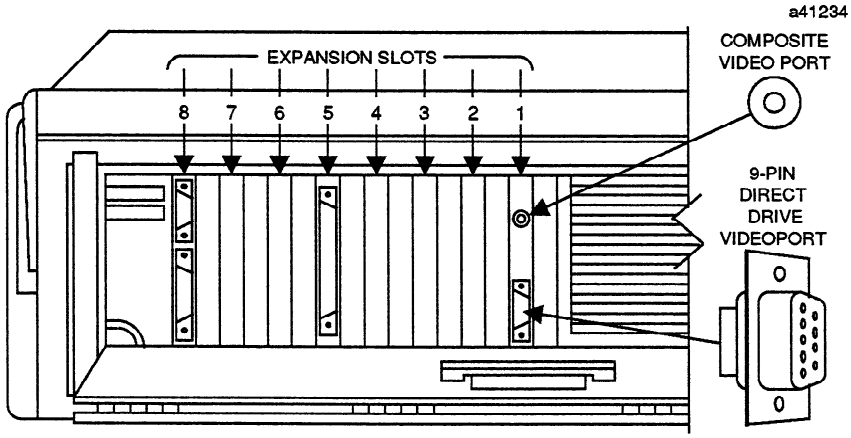


A CONNECTOR: CANNON DC37P (MALE)
B SCREWLOCK: CANNON D20419
C HOOD: CANNON DE110963-4
D CABLE: BELDEN 9300 SERIES (SHIELDED TWISTED PAIR)

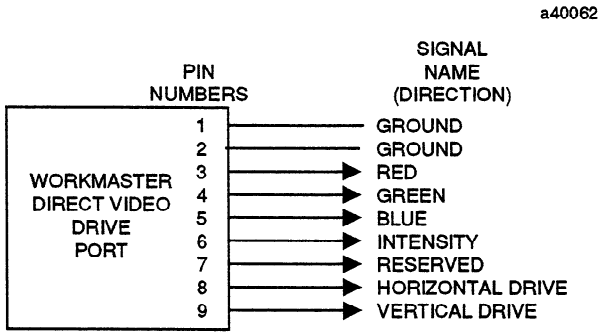
Color/Graphics Monitor Adapter Card

The Color/Graphics Monitor Adapter card (expansion slot 1 of the Workmaster computer) is connected internally to the amber monitor of the Workmaster unit.

Externally, it provides a direct drive TTL interface and a composite video interface. The direct drive interface uses a 9-pin D-type connector, and the composite interface uses a phono-plug type connector. The location of these interfaces is shown in the following illustration.

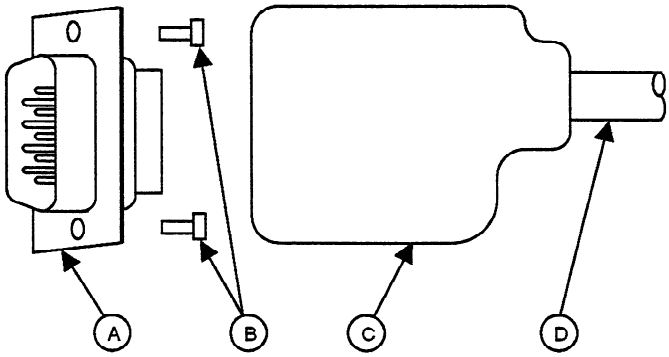


Pin-outs for the 9-pin connector for the direct drive port are shown below.



The following illustration shows the connector and cable for attachment to the Color/Graphics Monitor Adapter card.

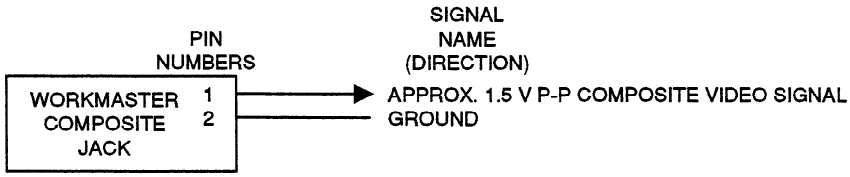
a40063



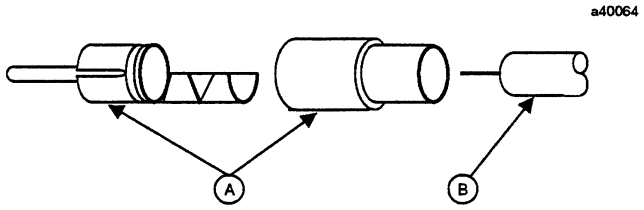
A CONNECTOR: CANNON DE9P (MALE)
B SCREWLOCK: CANNON D20419
C HOOD: CANNON DE110963-1
D CABLE: BELDEN 9300 SERIES (SHIELDED TWISTED PAIR)

The following illustration shows the circuits for the Composite Video Port:

a40066



The user connector and cable part numbers for the Color/Graphics Monitor Adapter card are shown below.



A CONNECTOR: SWITCHCRAFT 3558 PHONO PLUG
B CABLE: 75-OHM RG-58/U COAXIAL CABLE

Parallel Port Protocol

When communicating through the parallel port on the Workmaster computer, Logicmaster 6 software will make a character available on the data lines, wait for the printer to pull the BUSY (pin 11) line low, and then strobe the data to the printer by setting the STROBE (pin 1) line low and then high. This is repeated until the Logicmaster 6 system has no more characters to send to the printer.

Printer signals are made available through the parallel printer port on the back of the Workmaster computer. Signals and pin assignments are shown in the following table.

Pin Number	Signal Name	Description
1	Strobe	The printer may read the data bits when this signal is pulled low by the Logicmaster 6 software.
2	Data Bit 0	These signals represent bits 1 through 8 of the parallel data being sent to the printer.
3	Data Bit 1	
4	Data Bit 2	
5	Data Bit 3	
6	Data Bit 4	
7	Data Bit 5	
8	Data Bit 6	
9	Data Bit 7	
11	Busy	A character is not strobed to the printer as long as the printer holds this signal high.
10, 12-17		Not used.
18-25		Signal ground.

Serial Port Protocol

When sending a character to a printer through one of the serial ports on the Workmaster computer, Logicmaster 6 software first sets RTS (pin 4) and DTR (pin 9 or 20) high. It will then wait for CTS (pin 5) and DSR (pin 6) to be set high by the printer. Then, a character will be issued. This is repeated until the Logicmaster 6 system has no characters left to send to the printer.

If the printer wishes to prevent a character from being issued by the Workmaster computer, it must pull either CTS or DSR low before receiving the last data bit in the preceding character.

If a printer connected to one of the serial ports does not support this handshaking, the RTS signal pin must be jumpered to the CTS signal pin and the DTR signal pin must be jumpered to the DSR signal pin at the Workmaster side of the cable. The baud rate must then be adjusted so that the printer does not miss any characters.

Printer signals are made available through a 9-pin connector (port 1) and a 25-pin connector (port 2) on the back of the Workmaster computer. These signals and pin assignments are shown in the following table.

Pin Number	Signal Name	Description
2	TXD	Serial data output to the printer.
3	RXD	Serial data received from the printer.
4	RTS	This signal is set high by Logicmaster 6 software to indicate that it is ready to send data.
5	CTS	This signal is set high by the printer to indicate that it is ready to receive data.
6	DSR	This signal is set high by the printer to indicate that it is ready to receive data.
7	SG	Signal ground.
9*	DTR	This signal is set high by Logicmaster 6 to indicate that it is ready to send data.
20		

* Pin 9 on port 1; pin 20 on port 2.
Pins not used include 1 and 8 on port 1, and 1, 8-19, and 21-25 on port 2.

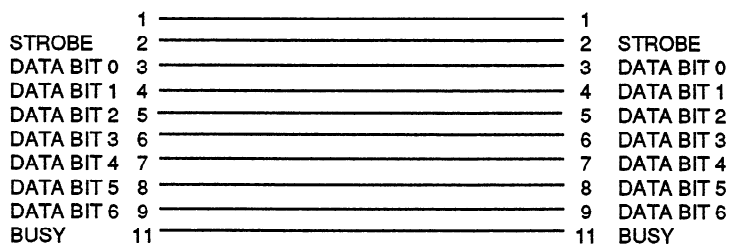
Printer Cable Diagrams

Cable diagrams for several types of printers are given below.

Parallel Interface with BUSY/STROBE Handshaking

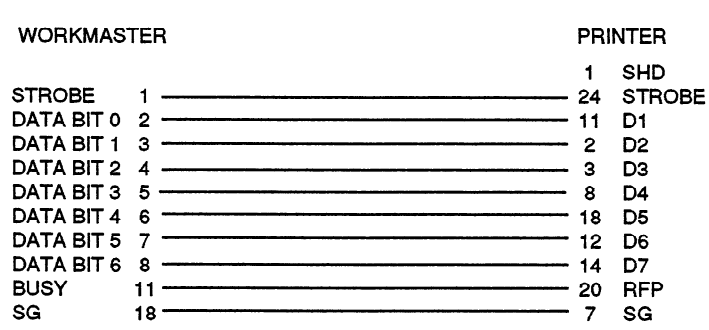
For IBM and Epson FX+ Series printers:

a42421



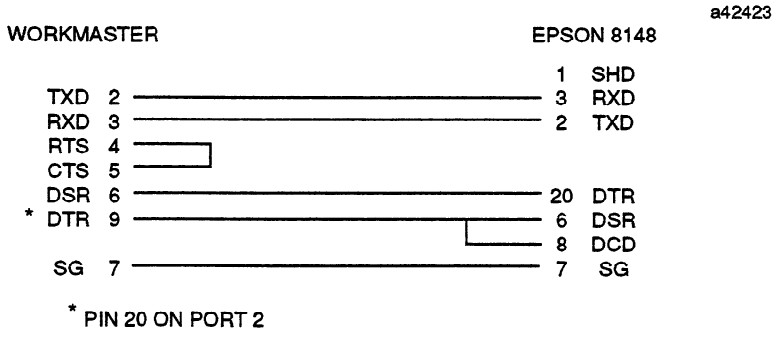
For TermiNet® 300 Series printers - standard parallel interface.

a42422

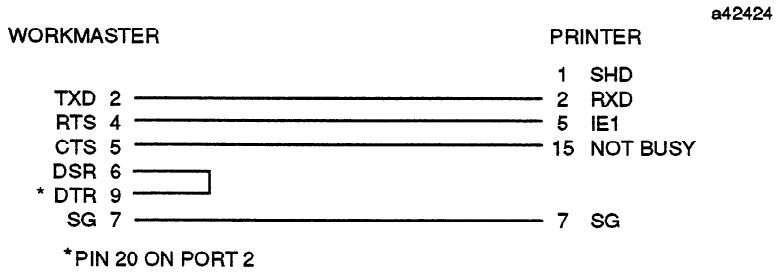


Serial Interface with Level One Handshaking

For Epson 8148 Intelligent Serial Interface (used with Epson FX and RX Series printers), Epson 8145 Serial Interface Type 2 (used with Epson MX Series printers):

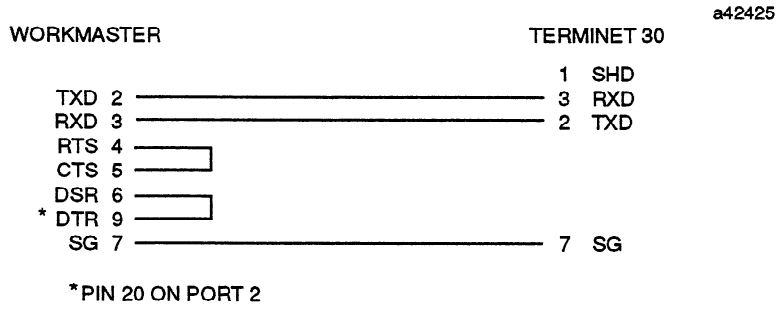


For Terminet 300 Series printers:



Serial Interface with Level Zero Protocol

For Terminet 30 printers:



Appendix B
Glossary of Terms

Address A specific memory location.

Advanced Functions A group of Series Six PLC program functions. The Advanced Functions include all basic functions, plus the ability to use auxiliary I/O references, signed double-precision arithmetic, subroutines, and other functions.

AND (Logical) A mathematical operation between bits. All bits must be 1 for the result to be 1.

Annotation Explanatory text in a program. Annotation includes names, nicknames, rung explanations, and coil labels.

ASCII American Standard Code for Information Interchange. An eight-bit (7 bits plus 1 parity bit) code used for data.

Background Some computer functions, such as background printing, can be performed in the "background" while other functions are being used.

Backup A duplicate version of a program, created prior to editing the program.

Basic Functions A group of program instructions including relays, timers, counters, shift-move, and basic arithmetic functions.

Baud A unit of data tramsmission. Baud rate is the number of bits per second transmitted.

Bit Binary Digit. This represents the smallest unit of data storage in memory. The value of a bit can be either 1 or 0.

Byte A group of 8 consecutive bits operated on as a single unit.

Byte Boundary The bit which marks the beginning of a new 8-bit byte. For example, 1,9,17, etc.

Chain A set of I/O channels. There are two I/O chains: the Main chain and the Auxiliary chain. In a conventional I/O system, each chain can have up to 1024 inputs and 1024 outputs. In an Expanded I/O system, each chain can have up to 8 channels (see below).

Channel In an Expanded I/O system, the term "channel" refers to a set of up to 1000 inputs and 1000 outputs. There can be up to 8 channels in the Main chain and 8 channels in the Auxiliary chain.

Constant A predetermined value stored in a register. This value does not change.

Counter A circuit internal to the PLC, which can be programmed to control other devices according to a preset number of on/off transitions.

CPU Central Processing Unit. The device that interprets instructions, makes decisions, and executes the instructions based on the decisions. In this manual, the term CPU is used to refer to the CPU module of the Series Six PLC.

CPU Keyswitch The upper keyswitch on the CPU of the Series Six PLC. Used to place the CPU in Run mode or Stop mode.

CPU Version The Series Six CPU contains operating instructions, called "firmware". The version of firmware in the CPU determines which program functions the CPU can perform.

Cross-Reference Table
A table that keeps track of program references.

Directory A file which contains the names and specifications of other files.

Discrete Refers to the inputs and outputs in the system. The term "discrete" includes both real and internal I/O.

Disk A hard disk or floppy diskette, used as an information storage and retrieval device.

DOS Disk Operating System. A group of utility programs which provide the structure for system operations.

Double Left Rail
The graphic representation of an area of ladder logic, the execution of which is under the control of an MCR function.

Double Precision Number
A twos complement value. In the Series Six PLC, it consists of two 16-bit words (31 bits plus the sign bit). The value range for such a number is -2,147,483,648 to +2,147,483,647.

Drive A floppy-diskette drive or hard disk drive. The identification of the drive, such as Drive A.

EOR (Exclusive OR)
A mathematical operation between 2 bits. If only one of the bits is 1, the result is 1. If the bits are both 0 or both 1, the result is 0.

Expanded Functions
A group of program functions available with the Series Six Plus PLC. The Expanded Functions include all other functions, plus the use of additional registers and I/O, floating point arithmetic, and other enhancements.

Explicit Reference
A register or I/O reference whose address is explicitly used as a progam reference.

Extended Functions
An earlier version of the Advanced Functions.

Firmware A series of software instructions contained in ROM (Read Only Memory). These instructions control internal operations.

Full Duplex A method of data transmission. In full-duplex transmission, data may be sent and received in both directions, simultaneously.

Function Key A key (F1-F8) whose function is controlled by software. This function which may change within the program. The Logicmaster 6 software displays the current assignments of the function keys at the bottom of the screen.

Genius I/O Block
A module which interfaces physical devices with the bus controller in the Series Six PLC. Blocks in a system communicate with the bus controller via a serial bus.

Genius I/O An intelligent I/O system consisting of I/O blocks, bus controllers, and other devices.

Half-Duplex A method of data transmission. In half-duplex transmission, data can only be sent in one direction at a time.

Help Screens Instructive text screens, displayed by pressing the Help (F10) key.

Implicit Reference
A register or I/O reference other than the explicit reference which is also used by a program function. For instance, in a table the first reference is the explicit reference, and the rest of the references are implicit.

IOR (Inclusive OR)
A mathematical operation between bits. If any bit is 1, the result is 1.

Input Devices Devices that mechanically or electrically supply data to the Series Six PLC. Typical input devices are limit switches, pushbuttons, pressure switches, digital encoders, and analog devices.

Instruction Set A group of program functions available for the Series Six or Series Six Plus CPU.

I/O Scan The CPU's monitoring of all inputs and all outputs within a prescribed time.

Internal Reference
A program reference that does not represent a hardware device.

INV (Logical Invert)
A mathematical operation on bits in a matrix. All ones are replaced by zeros, and all zeros are replaced by ones. The results are placed in another matrix.

Ladder Diagram
A graphic representation of decisional logic.

Latch A program function that causes an output to go on and stay on, even if power to the input is removed. It is a "retentive" function.

List A group of consecutive storage locations in memory, used for data manipulation. The beginning address and length of the list are set up in the program. Data is accessed from either the top or the bottom of the list.

Load The function used to transfer programs to the Logicmaster system's RAM memory.

Master Software
The original Logicmaster 6 software diskettes shipped from the factory.

Matrix A group of consecutive 16-bit storage locations in memory. The beginning address and length of the matrix are set up in the program. Individual bits in the matrix may be operated on by program instructions.

Memory Mapping
The assignment of inputs, outputs, and other data to pre-defined locations in CPU memory.

Memory Size The number of registers of memory in the CPU.

Millisecond (msec)
One thousandth of a second (0.001 second).

Mnemonic An abbreviation or other representation of a program instruction. The mnemonic appears in the ladder diagram where the function is used.

Mode Select Switch
The keyswitch on the front of the Workmaster computer that selects the mode of the Logicmaster 6 system. When using another type of computer, mode is selected in the software.

Monitor Mode A mode of operation that allows the operating program to be monitored. No program changes can be made in Monitor mode.

OR (Logical) A mathematical operation between bits, whereby if any bit is a 1, the result will be a 1.

Off-Line Mode A mode of operation used for program entry and editing, before the program is transferred to the CPU. This mode can be used for program development in a location remote from the CPU.

On-Line Changes Changes to I/O or register references, and certain other changes, made when the Logicmaster 6 system is on-line to an operating CPU, and the programs in both are exactly the same.

On-Line Mode A mode of operation that allows observation of an operating program. Certain changes may be made to the program while it is operating.

Override To remove control of an relay reference from its normal source. For instance, overridden relay inputs ignore information from input devices such as pushbuttons or limit switches.

Parallel Version Logicmaster 6 software that communicates with the Series Six CPU via an I/O cable to the I/O rack.

Parity A type of integrity check on data.

PLC A commonly-used abbreviation for Programmable Logic Controller.

Power Flow In a ladder diagram, the symbolic flow of power represents the logical execution of program functions. For each function, it is important to know what happens when power is received, and under what conditions power flow is output.

Rail The symbolic connection between ladder rungs. The left rail represents the positive power source.

RAM Random Access Memory. In this manual, the term RAM is used to refer to the volatile memory of the computer. This memory stores the Logicmaster software, program files, and related data while power is applied to the system.

Real Reference A program reference that represents a hardware device.

Reference An I/O or register address that supplies status or data to an instruction in the program.

Reference Tables A group of formatted tables which can display the values of I/O and registers in the system.

Register A group of 16 consecutive bits in register memory. Each register is numbered, beginning at 0001. Register memory is used for temporary storage of numerical values, and for bit manipulation.

Retentive Output An output that will remain on in its last state, even if power is removed.

Rung A unit of ladder logic. One rung may have up to 7 parallel lines of logic connected to the left rail, but these must combine so that there is just one connection to the right rail.

Scan The CPU's repeated execution of all program logic, I/O service, peripheral service, and self-testing. This occurs automatically, many times each second.

Scratch Pad A memory storage area in the PLC, which stores the characteristics of the CPU. A similar function in the Logicmaster 6 software is also called the Scratch Pad.

Serial Version Logicmaster 6 software that communicates with the Series Six CPU through a serial link to the CCM2 card, using CCM protocol.

Side File A secondary ladder logic file, consisting of part of a ladder logic program. This file can be added to another program.

Status Line The line at the top of the screen that shows the status of the CPU, the Logicmaster 6 mode, and other information.

Store The function used to transfer programs from the Logicmaster system's RAM memory to the CPU or to disk.

Supervisor Menu The main menu in Logicmaster 6 software. It lists all the principal system functions, and the function keys that control those functions.

Table A group of consecutive storage locations in memory. The beginning address and length of the table are specified in the program. Data may be accessed randomly in a table.

Teach Mode A function used to create customized key assignments for the F1 to F8 keys.

Timer An internal function that can be used to control the operating cycle of other devices by a preset and accumulated time interval.

Twos Complement A form of binary arithmetic used to perform binary subtractions with addition techniques. The twos complement negative of a binary number is formed by inverting each bit in the number and adding 1 to the result. For example, the twos complement of 0011 is 1100+1 or 1101.

Verify A function used to compare program content. The program in system RAM memory may be compared with a program from the CPU or from a disk drive.

View Mode A playback display of the key functions defined in Teach mode.

Watch Dog Timer A built-in timer which shuts down the CPU if the scan takes too long.

Word A group of 16 consecutive bits in the Input or Output tables.

Work Area The data-entry display in the lower right corner of the screen. The work area has three lines: the text (top) line, the reference (center) line, and the value (bottom) line.

Appendix C
Keyboard Translator Chart

This appendix contains a keyboard translator chart to use with the IBM-PC, PC-XT, PC-AT, or IBM-compatible personal computer. The chart has been printed in triplicate to provide you with extra copies. The sheet has also been formatted so that you can remove each copy of the chart from the manual for easier reference.

This page intentionally left blank.

Appendix D
Reference Used Tables

There are many ways to reference the same physical point. For example, AO0017 and R0002 or a program function using R0001 with a length of 2 registers. Therefore, it is easy to mistakenly use the same point for more than one reference.

The Reference Used tables indicate such conflicts by placing the letter C on any reference whose physical point is also used by another program reference.

On the following pages, three listings show the characters that may appear in the Reference Used tables. The types of usage are:

- Explicit Auxiliary
- Explicit Expanded
- Explicit Register
- Implicit Auxiliary
- Implicit Expanded
- Implicit Register

Table G-1. Auxiliary References Used Table

Symbol in the Table	Reference Used in the Program		
	Auxiliary	Expanded	Register
-	-	-	-
R	-	-	explicit
+	-	-	implicit
E	-	explicit	-
C	-	explicit	explicit
C	-	explicit	implicit
+	-	implicit	-
C	-	implicit	explicit
C	-	implicit	implicit
*	explicit	-	-
C	explicit	-	explicit
C	explicit	-	implicit
C	explicit	explicit	-
C	explicit	explicit	explicit
C	explicit	explicit	implicit
C	explicit	implicit	-
C	explicit	implicit	explicit
C	explicit	implicit	implicit
+	implicit	-	-
C	implicit	-	explicit
C	implicit	-	implicit
C	implicit	explicit	-
C	implicit	explicit	explicit
C	implicit	explicit	implicit
C	implicit	implicit	-
C	implicit	implicit	explicit
C	implicit	implicit	implicit

Table G-2. Definition of Symbols

Symbol	Definition
C	Conflict.
E	Explicit Expanded.
R	Explicit Register.
+	Any implicit use.
*	Auxiliary Explicit use. (Auxiliary Reference Used table)
-	Not used.

Table G-3. Expanded References Used Table

Symbol in the Table	Reference Used in the Program		
	Expanded	Auxiliary	Register
-	-	-	-
R	-	-	explicit
+	-	-	implicit
E	-	explicit	-
C	-	explicit	explicit
C	-	explicit	implicit
+	-	implicit	-
C	-	implicit	explicit
C	-	implicit	implicit
*	explicit	-	-
C	explicit	-	explicit
C	explicit	-	implicit
C	explicit	explicit	-
C	explicit	explicit	explicit
C	explicit	explicit	implicit
C	explicit	implicit	-
C	explicit	implicit	explicit
C	explicit	implicit	implicit
+	implicit	-	-
C	implicit	-	explicit
C	implicit	-	implicit
C	implicit	explicit	-
C	implicit	explicit	explicit
C	implicit	explicit	implicit
C	implicit	implicit	-
C	implicit	implicit	explicit
C	implicit	implicit	implicit

Table G-4. Definition of Symbols

Symbol	Definition
C	Conflict.
E	Explicit Expanded.
R	Explicit Register.
+	Any implicit use.
*	Expanded Explicit use. (Expanded Reference Used table)
-	Not used.

Table G-5. Register References Used Table

Symbol in the Table	Reference Used in the Program		
	Expanded	Auxiliary	Register
-	-	-	-
R	-	-	explicit
+	-	-	implicit
E	-	explicit	-
C	-	explicit	explicit
C	-	explicit	implicit
+	-	implicit	-
C	-	implicit	explicit
C	-	implicit	implicit
*	explicit	-	-
C	explicit	-	explicit
C	explicit	-	implicit
C	explicit	explicit	-
C	explicit	explicit	explicit
C	explicit	explicit	implicit
C	explicit	implicit	-
C	explicit	implicit	explicit
C	explicit	implicit	implicit
+	implicit	-	-
C	implicit	-	explicit
C	implicit	-	implicit
C	implicit	explicit	-
C	implicit	explicit	explicit
C	implicit	explicit	implicit
C	implicit	implicit	-
C	implicit	implicit	explicit
C	implicit	implicit	implicit

Table G-6. Definition of Symbols

Symbol	Definition
C	Conflict.
E	Explicit Expanded.
R	Explicit Register.
+	Any implicit use.
*	Register Explicit use. (Register Reference Used table)
-	Not used.

Appendix E
Software Function Key Flow Diagrams

The following diagrams illustrate the relationships between the software function keys. Position of functions within the diagram does not always represent actual key sequence.

Key function availability depends on the conditions listed below. If a key function does not display, check this list:

- Available software options.
- Scratch Pad content.
- Cursor position.
- Program logic at the cursor position.
- Logicmaster 6 operating mode (computer keyswitch position).
- CPU Memory Protect keyswitch position.
- Instruction set selected.

a40353/1

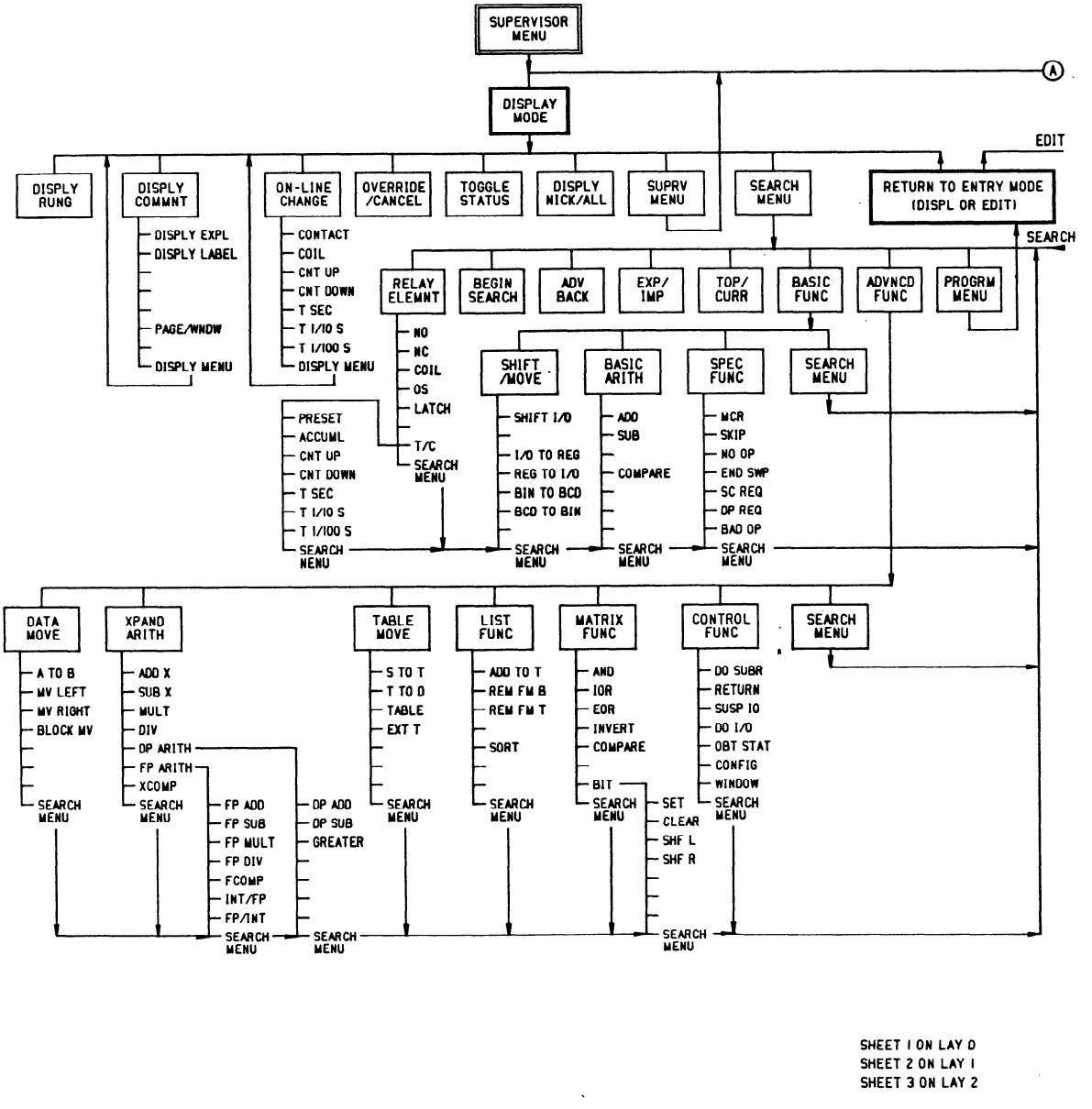


Figure H-1. Display Program Software Functions

a40353/2

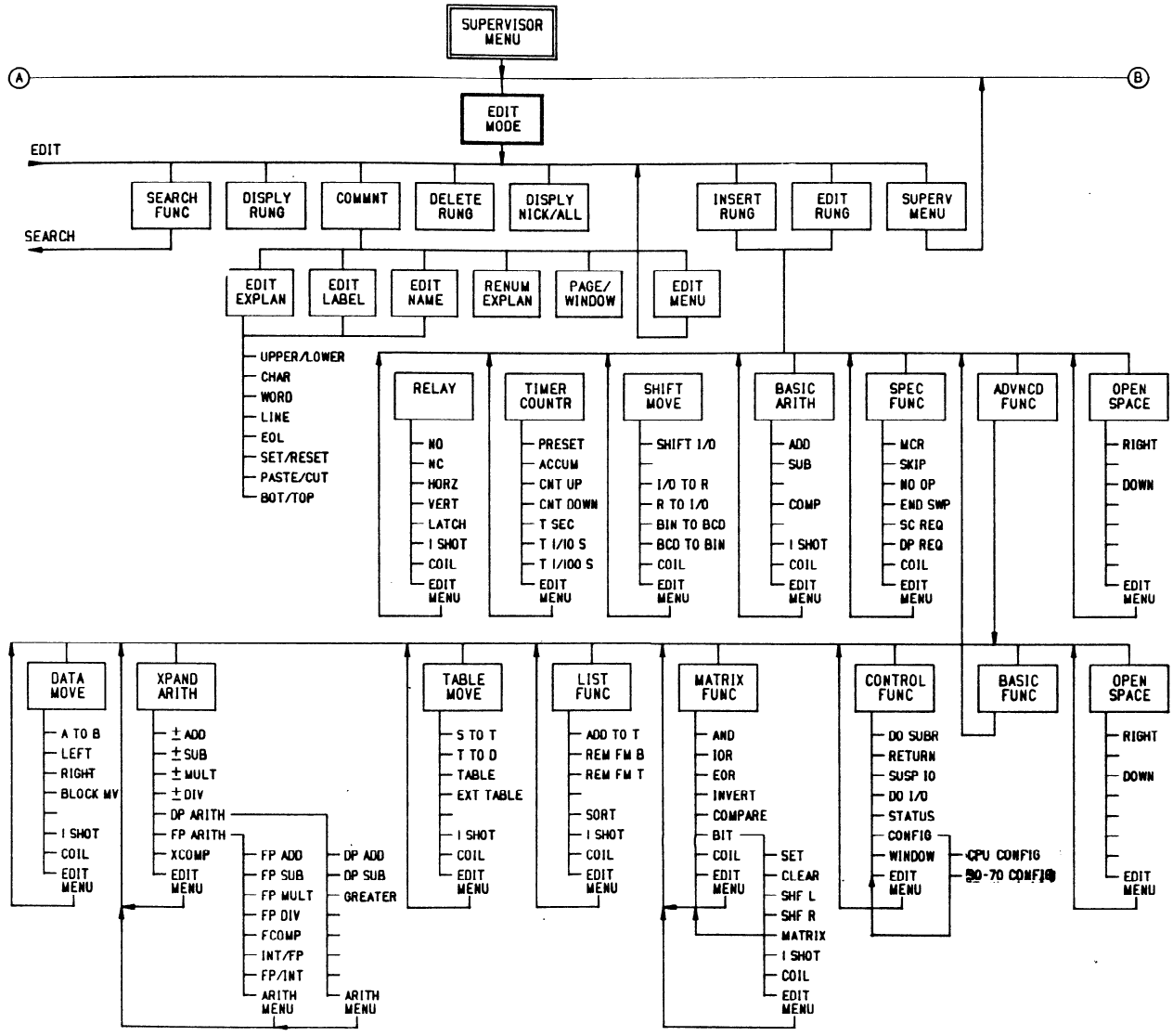


Figure H-2. Edit Program Software Functions

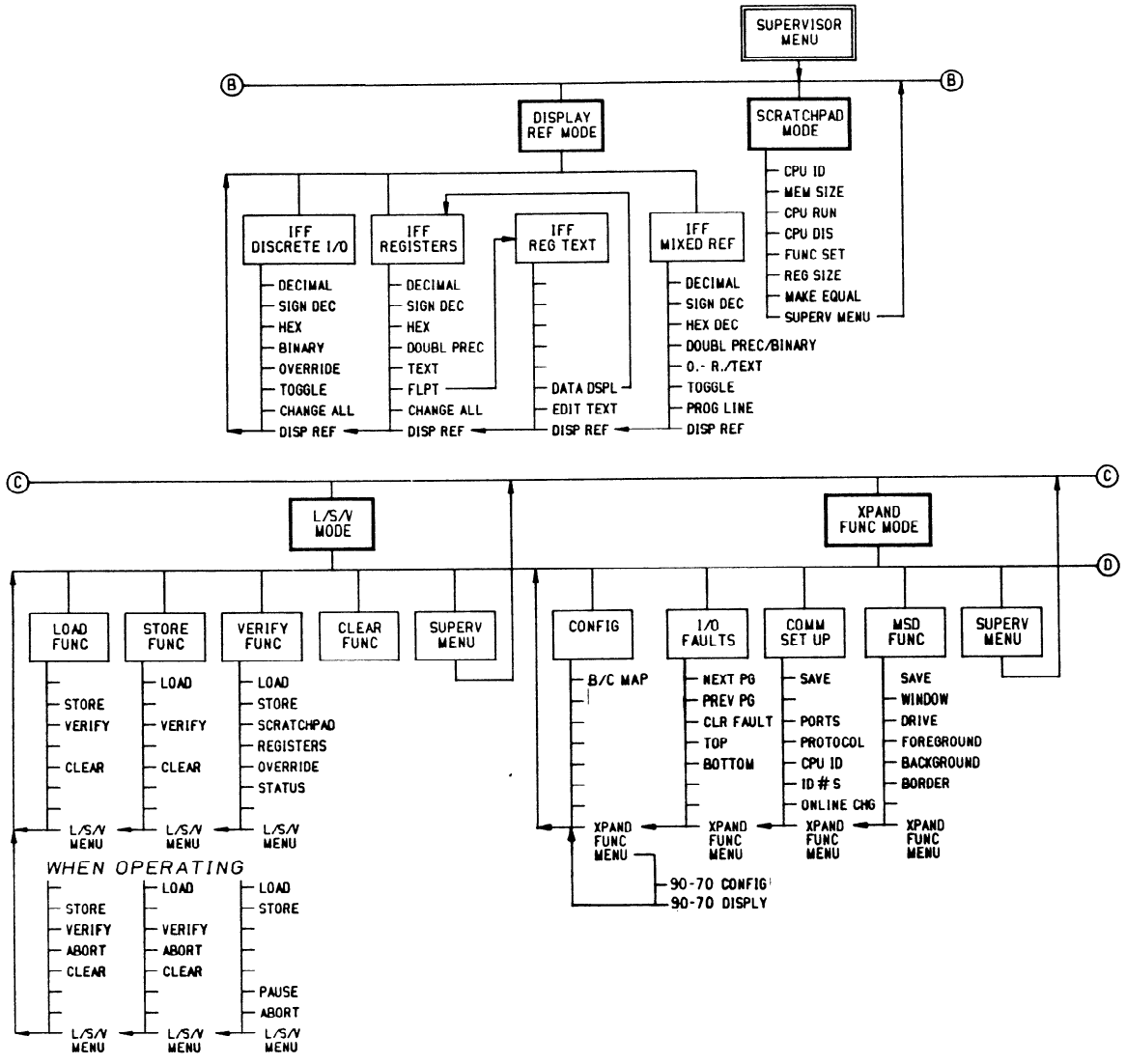


Figure H-3. Other Supervisor Menu Software Functions

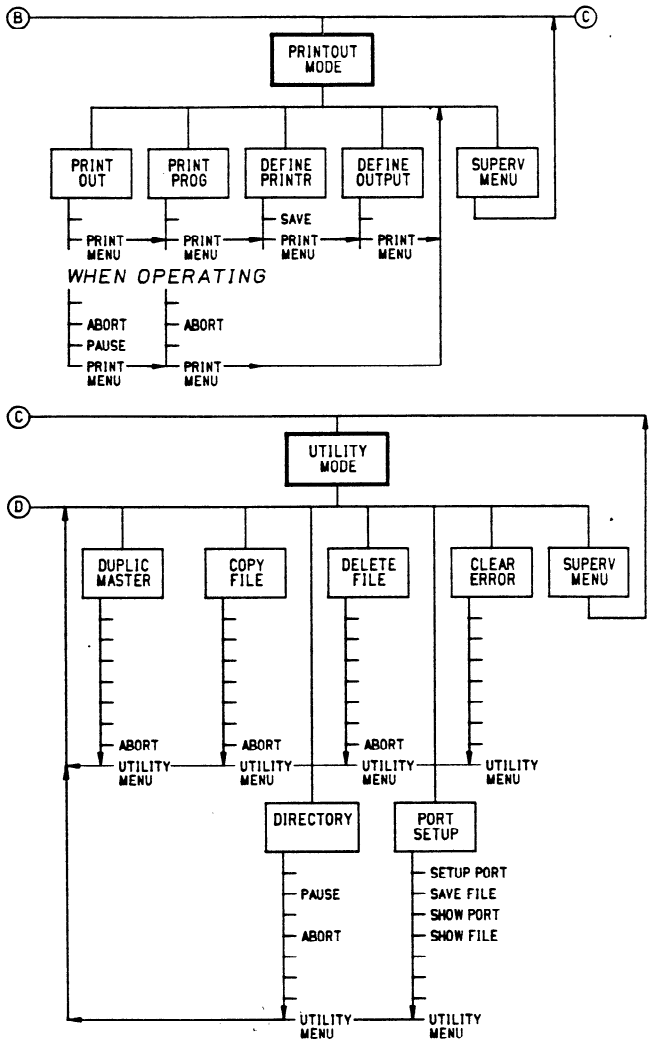


Figure H-4. Other Supervisor Menu Software Functions (Continued)

Appendix F
Error Codes

Function Level	Version	Error Number	Description
			Software Initialization Errors
All (Bundled)	All	6	Bad thermal switch.
All (Bundled)	All	14, 539 and 206	Check the following: 1. Faulty combination card. 2. Keyswitch bad or between positions. 3. Bad or disconnected thermal switch. 4. Combo card set as COM2 with no COM1. 5. Combo card cable is disconnected.
4.xx	All	5 and 6	Make sure that another software package is not already loaded into the system. Check for "mouse" drivers in the CONFIG.SYS and/or AUTOEXEC.BAT files. May also be caused by a large Path statement in the AUTOEXEC.BAT file, exceeding the system's environment area. A DOS Shell command can make this area larger. Correct and reboot the system.
All	All	2	Out of RAM memory. This message appears during power-up. Check the AUTOEXEC.BAT file for a large Path statement. Use the DOS Shell command to make this area larger, or delete some Path statements. Check the CONFIG.SYS file for "VDISK" assignment. Check the system for at least 640K memory.
			System Software Errors
All	All	000 and 099	Not enough RAM memory to run the software. This message appears during power-up. Perform the same checks described for software initialization error 2, described above.
4.01 4.02 4.03	Parallel and Serial Parallel and Serial Serial	405	If you delete digits of a hex number in the numerical work area of the register reference table, the work area defaults to a to a strange number base. When the Enter key is pressed, error 405 is displayed.
4.xx	All	226	False Write Protect Error. This error occurs when setting the 64K window size. There is a problem with the Machine.Set file on Logicmaster 6 disk 1 of 1. Delete the Machine.Set file using DOS, and reboot the computer.

Function Level	Version	Error Number	Description
		Other	
4.03	All	None	Programmer lock-up when scrolling. The computer may lock up when scrolling up or down in Display mode, with annotation active. This problem is related to the size of the buffer and file allocation in the CONFIG.SYS file and other drivers. Check for correct values in the CONFIG.SYS file, and reboot the programmer.
4.xx	All	None	Annotation file not found; attempting to print ladder diagram with annotation. Check the Define Output screen, and select N (for NO) for each item until the ladder begins printing. Then, restore other items and proceed with the printout.

Appendix G
Annotation Files

This appendix describes annotation file structures for release 4 of the Logicmaster 6 software. (Annotation file structures for release 3 of the software are the same.) The information is provided as an aid to programmers creating customized documentation like wire lists, I/O assignment tables, and I/O rack drawings. Such programs should run on a Workmaster or Cimstar I industrial computer, or on one of the IBM personal computers or an equivalent.

Annotation files are complex structures. The programmer must be familiar with databases and handling binary files.

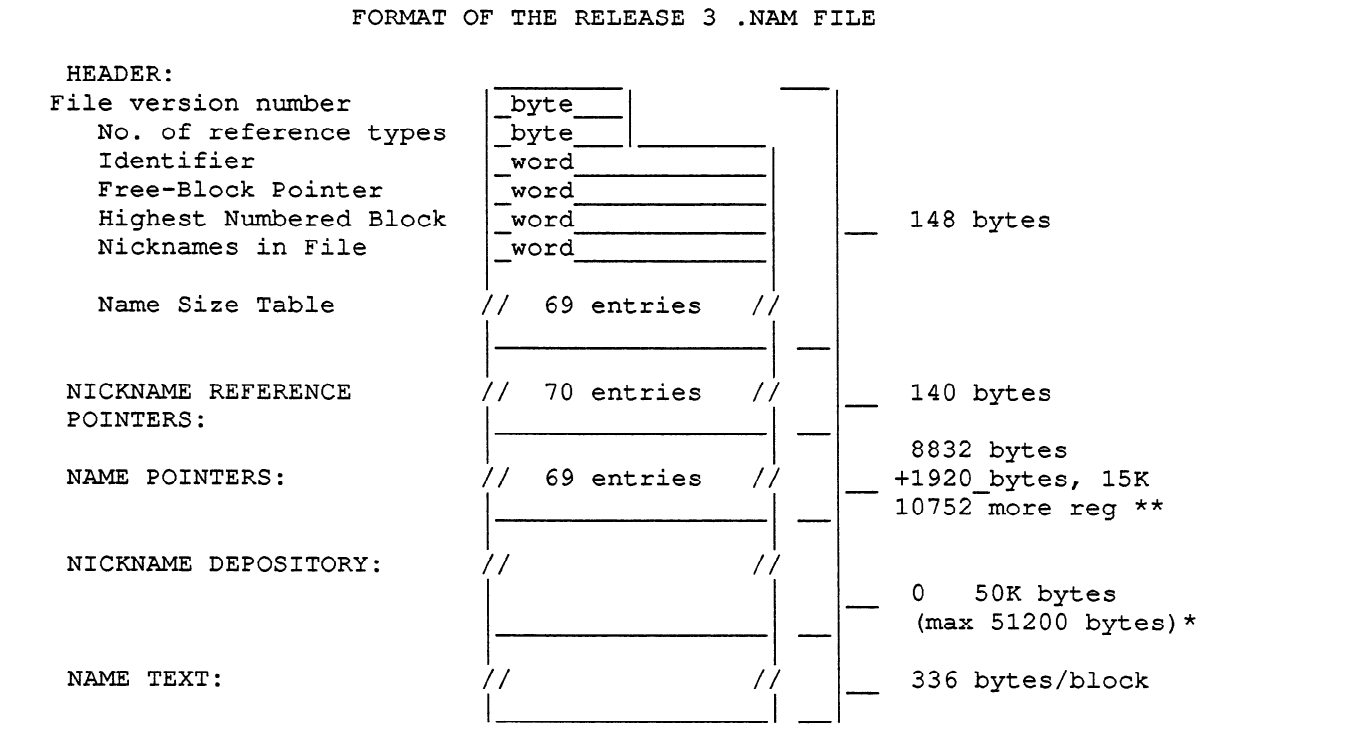
When annotation is selected for a ladder logic program, two file structures are created. These files have the same name as the program itself, and are intended for use with that particular program. The files are identified by the extensions .NAM and .EXP to the main file name.

For example, if program1 is the name of the program, then program1.NAM is the Names/Nicknames file. It is referred to as the .NAM file. The .NAM file contains all reference nicknames and names for a ladder logic program. The format and content of the .NAM file are described below.

If program1 is the name of the program, then program1.EXP is the Rung Explanations/Coil Labels file. It is referred to as the .EXP file. The .EXP file contains coil labels and rung explanation text for a ladder logic program. The format and content of the .EXP file are described later in this appendix.

.NAM File

The .NAM file consists of a header, a set of pointers for the Nickname Depository, a set of pointers to blocks of Name text, the Nickname Depository, and blocks of Name text that have been created.



*10 bytes per Nickname.

** (For all references +1K registers)
(For 1K registers)

8832 bytes = (69 groups of 1K references) X (64 pointers/1K references) X2

1920 bytes = (15K register references) X (64 pointers/1K references) X2

10752 bytes for all references + 64K registers)

.NAM File Header

The .NAM file header consists of:

1. File version number (1 byte).

   The file version number is:

LM6 Release	Version
1.02	1
1.03	2
2.01	3
3.01	4
4.01	4

2. Number of reference types supported by the .NAM file (1 byte).

   This number shows how many program reference types (I, O, R, AI, AO, O External, O Internal, I External, I Internal) are currently supported (for nickname and name assignments) in the .NAM file. The value in the .NAM file for release 4 of the Logicmaster 6 software is 69. That corresponds to the following reference types:

Reference Type	Description
I	Input
O	Output
R	Register
AI	Auxiliary input
AO	Auxiliary output
O	External Channels 0-F
O	Internal Channels 0-F
I	External Channels 0-F
I	Internal Channels 0-F

3. Unused (1 word)
4. Free-block pointer (1 word)

   The free-block pointer is used like a Name-block pointer to reference blocks of Name text (for more information, refer to the description of Name-block pointers). A linked list of previously-used, but now empty, Name text blocks is maintained. The free-block pointer indicates the first Name text block in that list.

   The pointer to the second previously-used Name text block is in the first two bytes of the first Name block on the free list. The pointer to the third previously-used Name text block is in the first two bytes of the second Name block, and so on.

   The last Name text block in the list has a nil pointer (FFFF hex) in the first two bytes of the block. If there are no previously-used Name text blocks in the file, the free-block pointer is a nil pointer (FFFF hex).

5. Highest block number (1 word)

   The block number of the last block in the .NAM file is the highest block number. It is one less than the number of Name text blocks that follow the Nickname Depository area (because the number of the first Name text block is 0). For example, in a file with 7 Name text blocks, the highest Name text block is number 6. This is true whether or not the file has a Nickname Depository area.

   The highest block number never decreases. It increments as new Name text blocks are created at the end of the file. Even if all the Name text blocks in the .NAM file are on the list of unused blocks, the highest block number remains the same because it indicates physical file size. If a .NAM file has no Name text blocks, the highest block number is the nil pointer value (FFFF hex). In this case, creating one Name text block causes the highest block number to be 0. Any other Name text blocks created at the end the file cause the highest block number to increment accordingly.

6. Number of Nicknames in the .NAM File (1 word)

   This value indicates the number of nicknames currently supported, given the size of the Nickname Depository area, in the .NAM file. The following table indicates the number of nicknames actually in the Nickname Depository and the number of nicknames stored in this word of the .NAM file header:

Number of Nicknames in Depository	Value Stored in .NAM File Header for Number of Nicknames
# = 0	0
1 < # < 1025	1024
1024 < # < 2049	2048
2048 < # < 3073	3072
3072 < # < 4097	4096
4096 < # < 4121	5120

   The number of nicknames stored in the .NAM file header never decreases as the number of assigned nicknames in the Nickname Depository decreases. It does increase as the number of assigned nicknames in the depository increases. Therefore, the disk area occupied by the depository can only grow. It can never shrink.

7. The Name Size Table

   This table contains 69 one-word entries. Their values indicate, in multiples of 1024, how many addresses for each reference type the .NAM file currently supports with Name block pointers. The Name Size table for Release 4 of the Logicmaster 6 software contains the following:

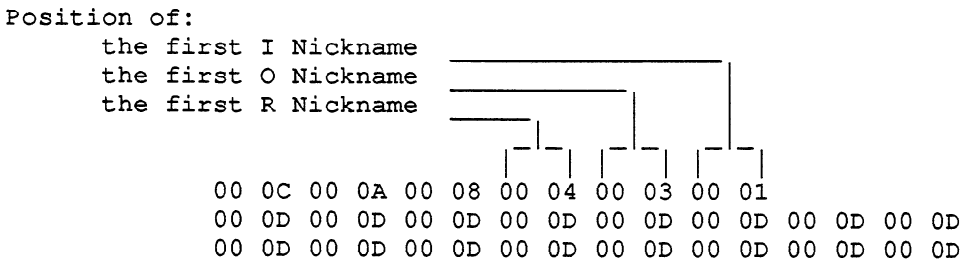
Reference Type	Table Contents Multiple of 1024
I	1
O]	1
R	16
AI	1
AO	1
O0+	1
I0+	1
O1+	1
I1+	1
.	.
.	.
Of+	1
If+	1
O0-	1
I0-	1
O1-	1
I1-	1
.	.
.	.
Of-	1
If-	1

Nickname Reference Pointers

There are 70 Nickname reference pointers. This corresponds to the 69 reference types. Each pointer occupies one word in the file. The order of the Nickname reference pointers is:

I
O
R
AI
AO
O0+
I0+
O1+
I1+
↓
Of+
If+
O0-
I0-
O1-
I1-
↓
Of-
If-

The entry for each pointer indicates the relative position of the first nickname of that reference type in the Nickname Depository. The following three lines represent the beginning of an example group of Nickname Reference pointers:



The first input reference nickname occupies position 1 in the Nickname Depository. In this example, there are two input reference nicknames, so the first output reference nickname is at position 3.

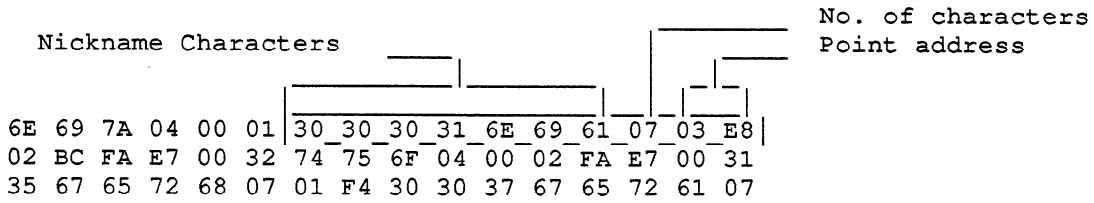
In this example, the position of the first channel 0 external output (O0+) is OD. Where subsequent reference types have no nicknames, their pointers are filled with the same value.

The last (70th) pointer indicates the end of the Nickname Depository. For a Workmaster computer with 640K RAM, the Nickname reference pointer for the end of depository (70th entry) cannot exceed 2K+1. For version of 3.02 or greater, you can have 5K+1 with windowing enabled in the MSD menu (see Expanded Functions).

Nickname Depository

Nickname reference pointers indicate the position of nicknames in the Nickname Depository. Nickname entries are arranged alphabetically.

Each entry is 10 bytes long and is read from right to left. The first two bytes of each nickname entry contain the reference address associated with a nickname. The third byte indicates the number of characters in the nickname. The remaining 7 bytes contain the characters of the nickname. The following partial example shows the first three lines of a Nickname Depository. The first entry is for an input reference. In the example, a box indicates the 10 bytes for the first entry:



In this example, the first 10 bytes of the Nickname Depository represent a nickname for input I1000. The first two bytes represent the point address (1000), converted to hexadecimal. The third byte means the Nickname has 7 characters. The remaining 7 bytes contain hexadecimal representations of the nickname characters.

The following example shows nickname assignments with a set of reference type pointers and the corresponding Nickname Depository:

Nickname Assignments
zin1 = I 0001
ain1000 = I 1000
out2 = O 0002
hreg500 = R 0500
qreg600 = R 0600
areg700 = R 0700
reg1024 = R 1024
auxin5 = AI 0005
auxin7 = AI 0007
aout100 = AO 0100
aout10 = AO 0010
out0+25 = O0+ 0025
inf+100 = If+ 0100
outf-44 = Of- 0044
in5-555 = I5- 0555

Reference Type	Nickname Reference Pointer in .NAM File
I	1
O	3
R	4
AI	8
AO	10
O0+	12
I0+	13
O1+	13
I1+	13
//	//
Of+	13
If+	13
O0-	14
I0-	14
01-	14
I1-	14
//	//
I5-	14
O6-	15
//	//
Of-	15
If-	16
end of depository	16

Nickname Depository		
Reference Address	**Nickname Length**	**Nickname**
1000	7	ain1000
1	4	zin1
2	4	out2
700	7	areg700
500	7	hreg500
600	7	qreg600
1024	7	reg1024
5	6	auxin5
7	6	auxin7
10	6	aout10
100	7	aout100
25	7	out0+25
100	7	inf+100
555	7	in5-555
44	7	outf-44

The Nickname Depository will occupy the following number of bytes on the disk:

Number of Assigned Nicknames in Depository	Depository Size in Bytes
# = 0	0
1 < # < 1025	10240
1024 < # < 2049	20480
2048 < # < 3073	30720
3072 < # < 4097	40960
4096 < # < 4121	51200

Name Block Pointers and Name Text Blocks

The Name block pointers (1 word each) point to Name text blocks of 16 names. Each Name text block contains the 21-character Name text for a range of 16 addresses. For example, a text block will contain the names for I0017 to I0032, in that order. With 16 names per Name text block and 1024 addresses for possible name assignments per reference type (I, O, R, AI, AO, external output channels 0-F, internal output channels 0-F, external input channels 0-F, internal input channels 0-F), 64 Name block pointers are required for each 1024 addresses. The number of Name block pointers for each reference type or the number of Name block pointers in the .NAM file can be determined by using the entries in the Name size table of the .NAM file header. These represent the number of K of addresses, which may have names, for each reference type. The entries of the Name Size table are to be used as multipliers of the 64 name-block pointers per 1K of reference addresses.

A Name block pointer is an offset, in units of Name text blocks, from the end of the Nickname Depository in the .NAM file. For instance, the first block of Name text in the file is located directly after the last byte of the Nickname Depository. Its Name block pointer is 0. The second Name text block follows the first block of Name text, which is 336 bytes in length (16 names by 21 characters/name). The Name block pointer for the second Name Text block is 1, indicating that it begins one Name Text block from the end of the Nickname Depository.

If a Name text block becomes empty, its pointer is added to the linked list of free Name Text blocks pointed to by the free-block pointer in the .NAM file header. If no Name text block exists for a group of 16 names, the Name block pointer for that group of 16 names will be a nil pointer (FFFF hex).

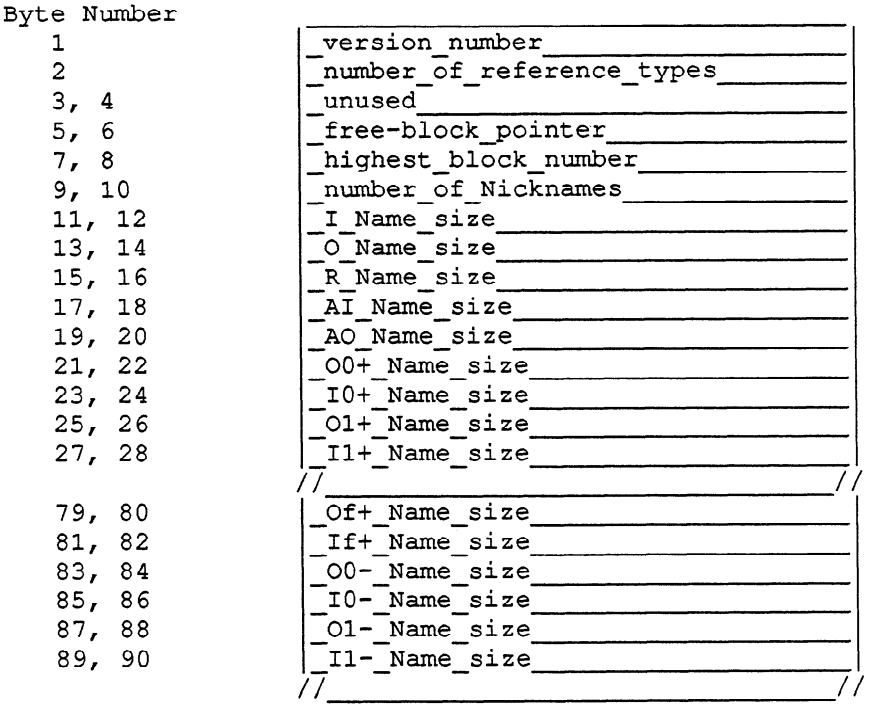
All information in the .NAM file is referenced by its byte number relative to the beginning of the file. The .NAM file header and Nickname reference pointers are of fixed size, so their entries may be referenced by fixed byte locations. The Name Size table is used to calculate either the size of all the Name block pointers or, with the fixed size of the .NAM file header and Nickname reference pointers, the offset from the beginning of the file of any particular reference type's Name block pointers. The size in bytes of the Nickname Depository can be calculated using the number of nicknames in the file from the .NAM file header and the known byte size of a depository entry (10 bytes). With the above information, the byte location of a block of name text can be calculated as follows:

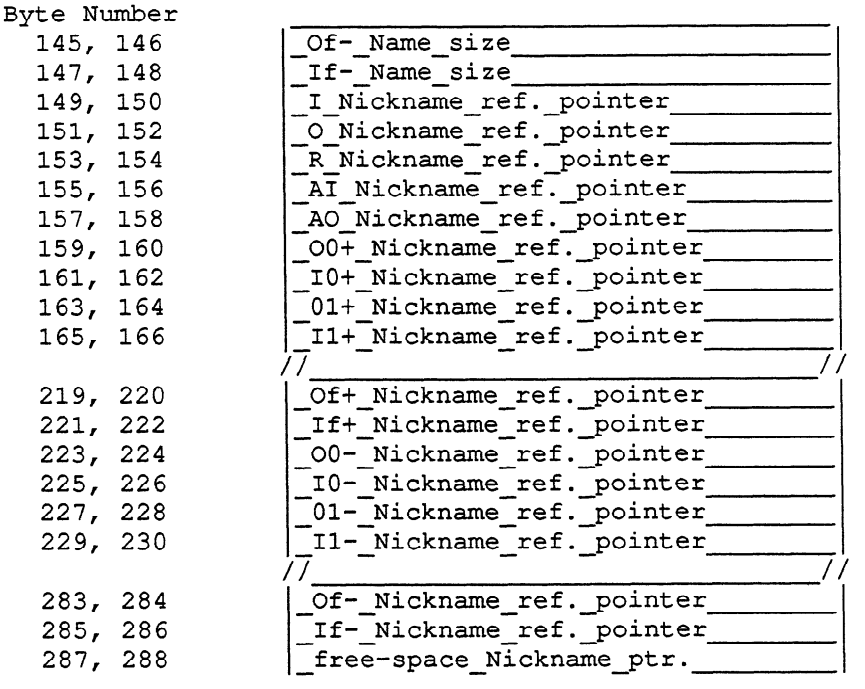
```
                    byte location of block of name text =

                                .NAM file header size (bytes) +
                nickname reference pointers size (bytes) +
                        name block pointers size (bytes) +
                        nickname depository size (bytes) +
(name block pointer  x  size of name text block) (bytes)
```

.NAM File Structure

The following pages show the structure of the .NAM file by byte number.

.NAM File Header:





Name Block Pointers:

Byte Number	
289-416	I_1_to_16_Names_pointer
	I_n_to_n+15_Names_pointer
	I_1009_to_1024_Names_pointer
417-544	O_1_to_16_Names_pointer
	O_n_to_n+15_Names_pointer
	O_1009_to_1024_Names_pointer
545-2592	R_1_to_16_Names_pointer
	R_n_to_n+15_Names_pointer
	R_16369_to_16384_Names_pointer
2593-2720	AI_1_to_16_Names_pointer
	AI_n_to_n+15_Names_pointer
	AI_1009_to_1024_Names_pointer
2721-2848	AO_1_to_16_Names_pointer
	AO_n_to_n+15_Names_pointer
	AO_1009_to_1024_Names_pointer
2849-2976	O0+_1_to_16_Names_pointer
	O0+_n_to_n+15_Names_pointer
	O0+_1009_to_1024_Names_pointer
2977-3104	O1+_1_to_16_Names_pointer
	O1+_n_to_n+15_Names_pointer
	O1+_1009_to_1024_Names_pointer
3105-3232	I1+_1_to_16_Names_pointer
	I1+_n_to_n+15_Names_pointer
	I1+_1009_to_1024_Names_pointer
3233-3360	I1+_1_to_16_Names_pointer
	I1+_n_to_n+15_Names_pointer
	I1+_1009_to_1024_Names_pointer

Byte Number	
6689-6816	Of+_1_to_16_Names_pointer
	Of+_n_to_n+15_Names_pointer
	Of+_1009_to_1024_Names_pointer
6817-6944	If+_1_to_16_Names_pointer
	If+_n_to_n+15_Names_pointer
	If+_1009_to_1024_Names_pointer
6945-7072	O0+_1_to_16_Names_pointer
	O0+_n_to_n+15_Names_pointer
	O0+_1009_to_1024_Names_pointer
10785-10912	Of-_1_to_16_Names_pointer
	Of-_n_to_n+15_Names_pointer
	Of-_1009_to_1024_Names_pointer
10913-11040	If-_1_to_16_Names_pointer
	If-_n_to_n+15_Names_pointer
	If-_1009_to_1024_Names_pointer

Nickname Depository:

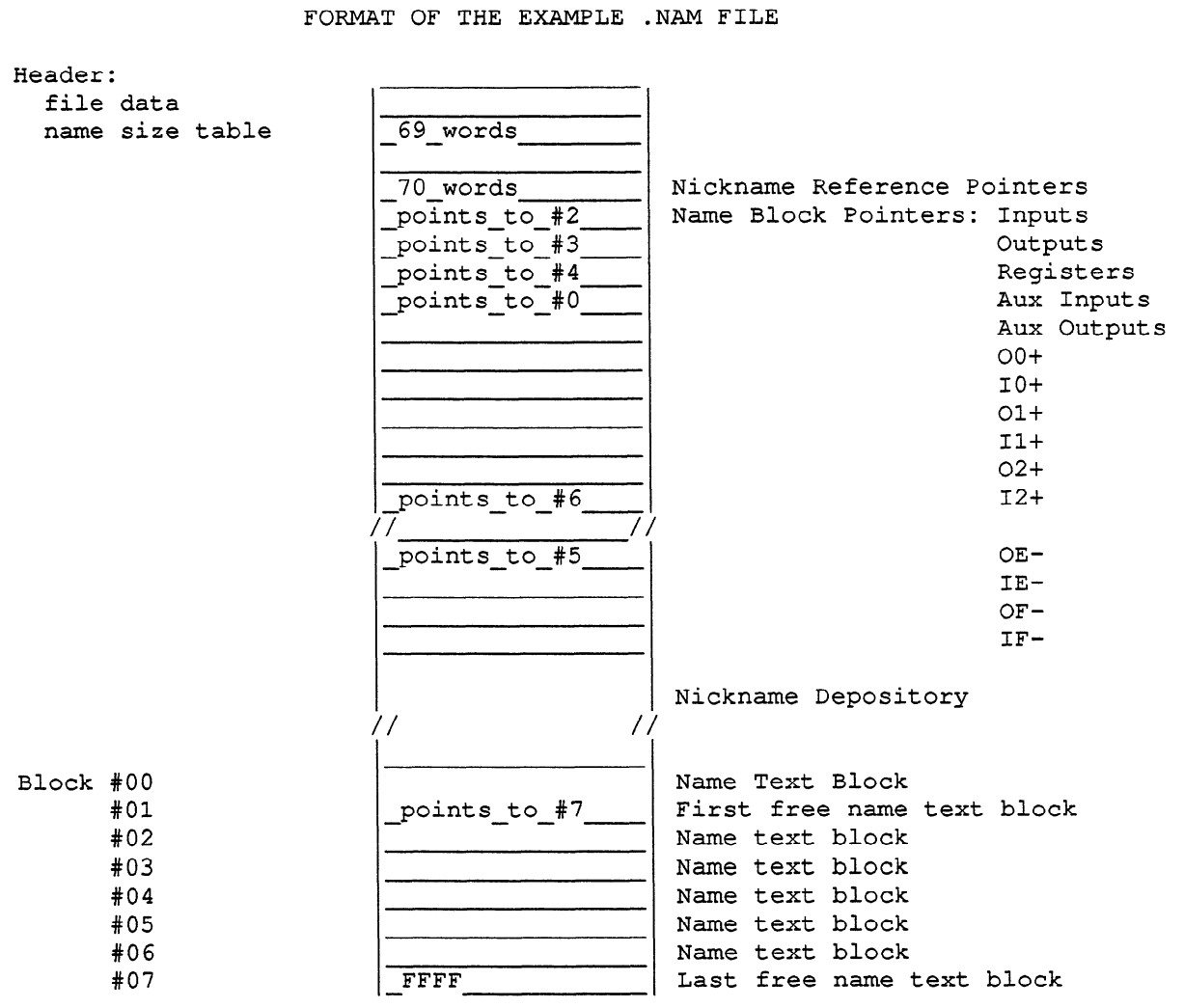
Byte Number	
Assuming space for 1K Nicknames	Nickname Depository
11041-21280	#_Nickname_entries

Name Text Blocks:

Byte Number	
21281-21616	text_for_16_Names
21617-21952	text_for_16_Names
21953-22288	text_for_16_Names
22289-22624	text_for_16_Names
22625-22960	text_for_16_Names
	more text, as needed

Example .NAM File

An annotated example .NAM file for release 4 Logicmaster 6 software appears on the following pages. The illustration below shows the basic arrangement of data in the example file as specified on page F-8.



NOTE

In the example file contents that follow, data and block numbers are in hexadecimal.

Example .NAM File Header

```
Highest block number _____                  _____________ Free Block Pointer
No. of Nicknames ____|                     |_____________ unused
                     |                     |_____       Number of ref. types
Name Size table  <|  |_|_| |_|_| |_|_| |_|_| |_| |__    Version number
                  | | | | | | | | | | | | | | |
  00 10 00 01 00 01 04 00 00 07 00 01 00 2A 45 04 ..............  000001
  00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 ..............  000011
  00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 ..............  000021
  00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 ..............  000031
  00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 ..............  000041
  00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 ..............  000051
  00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 ..............  000061
  00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 ..............  000071
  00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 ..............  000081
                                    00 01 00 01 ..............  000091
```

Example Nickname Reference Pointers:

```
  00 0C 00 0A 00 08 00 04 00 03 00 01             ..............  000095
  00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D ..............  0000A1
  00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D ..............  0000B1
  00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D ..............  0000C1
  00 0E 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D 00 0D ..............  0000D1
  00 0E 00 0E 00 0E 00 0E 00 0E 00 0E 00 0E 00 0E ..............  0000E1
  00 0F 00 0F 00 0F 00 0F 00 0F 00 0E 00 0E 00 0E ..............  0000F1
  00 0F 00 0F 00 0F 00 0F 00 0F 00 0F 00 0F 00 0F ..............  000101
  00 10 00 10 00 0F 00 0F 00 0F 00 0F 00 0F 00 0F ..............  000111
```

Example Name Block Pointers:

Input

In the example file, there is one Name block pointer (shown here in a box) for inputs. It contains the offset number (02) of the Name text block for inputs I0001 through I0016.

```
  FF FF FF FF FF FF FF FF FF FF FF FF FF FF|00 02|..............  000121
  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000131
  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000141
  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000151
  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000161
  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000171
  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000181
  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000191
```

Output

In this file, there is one Name block pointer for outputs. It contains the offset number (03) of the Name text block for outputs O0017 through O0032.

```
FF FF FF FF FF FF FF FF FF FF FF FF|00_03|FF FF ..............  0001A1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0001B1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0001C1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0001D1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0001E1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0001F1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000201
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000211
```

Register

In this file, there is one Name block pointer for registers. It contains the offset number (04) of the Name text block for registers R15983 through R16000.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000221
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000231
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000241
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000251
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000261
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000271
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000281
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000291
```

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0009A1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0009B1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0009C1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0009D1
|00_04|FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0009E1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0009F1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000A01
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000A11
```

Aux Input

There is one Name block pointer for auxiliary inputs. It contains the offset number (00) of the Name text block for auxiliary inputs AI0017 through AI0032.

```
FF FF FF FF FF FF FF FF FF FF FF FF|00_00|FF FF ..............  000A21
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000A31
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000A41
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000A51
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000A61
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000A71
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000A81
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000A91
```

Aux Output

There is no Name text for auxiliary outputs, so there is no pointer in this part of the file.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000AA1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000AB1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000AC1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000AD1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000AE1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000AF1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000B01
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000B11
```

Output Channel 0 External

There is also no Name text for output channel 0+ through output channel 2+.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000B21
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000B31
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000B41
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000B51
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000B61
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000B71
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000B81
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000B91
```

Input Channel 0 External

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000BA1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000BB1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000BC1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000BD1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000BE1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000BF1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000C01
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000C11
```

Output Channel 1 External

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000C21
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000C31
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000C41
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000C51
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000C61
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000C71
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000C81
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000C91
```

Input Channel 1 External

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000CA1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000CB1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000CC1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000CD1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000CE1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000CF1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000D01
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000D11
```

Output Channel 2 External

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000D21
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000D31
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000D41
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000D51
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000D61
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000D71
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000D81
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000D91
```

Input Channel 2 External

In this file, there is one Name block pointer for input channel 2+. It contains the offset number (06) of the Name text block for inputs I2+0033 through I2+0048.

```
FF FF FF FF FF FF FF FF FF FF|00 06|FF FF FF FF ................ 000DA1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000DB1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000DC1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000DD1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000DE1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000DF1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E01
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E11
```

Output and input channels O3+ through OD- are not shown for this example. Assume that there are no pointers used for those channels.

Output Channel E Internal

There is one Name block pointer for output channel E-. It contains the offset number (05) of the Name text block for outputs OE-0993 through OE-1008.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002921
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002931
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002941
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002951
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002961
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002971
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002981
FF FF|00 05|FF FF FF FF FF FF FF FF FF FF FF FF ..... 002991
```

Input Channel E Internal

There are no Name block pointers for input channel E- through input channel F-.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 0029A1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 0029B1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 0029C1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 0029D1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 0029E1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 0029F1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002A01
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002A11
```

Output Channel F Internal

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002A21
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002A31
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002A41
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002A51
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002A61
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002A71
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002A81
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002A91
```

Input Channel F Internal

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002AA1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002AB1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002AC1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002AD1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002AE1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002AF1
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002B01
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..... 002B11
```

Example Nickname Depository:

Each 10 bytes of the Nickname Depository represent the nickname for one reference. Each entry contains the reference point address, the number of characters, and up to 7 text characters. If fewer than 7 characters are needed, the 10 bytes are filled with trailing zeros. In the example Nickname Depository for this file, boxes are shown around the reference addresses. Each of these is the hexadecimal representation of the I/O point address. For example, 03E8 in the first box equals the point address 1000. The position of this entry in the Nickname Depository shows that the reference type is an Input. Therefore, this entry contains the nickname for I1000.

```
 6E_69 7A 04|00_01|30 30 30 31 6E_69 61 07|03_E8|...ain1000...zin 002B21
|02_BC|FA E7 00 32 74_75 6F 04|00_02|FA E7 00 31 1......out2..... 002B31
 35 67 65 72 68 07|01_F4|30 30 37 67 65_72 61 07 .areg700...hreg5 002B41
 72 07|04_00|30 30 36 67 65_72 71 07|02_58|30 30 00...qreg600...r 002B51
 FA 35 6E 69 78_75 61 06|00_05|34 32 30 31 67_65 eg1024...auxin5. 002B61
 75_6F 61 06|00_0A|FA 37 6E 69 78_75 61 06|00_07|...auxin7....aou 002B71
|00_19|30 30 31 74 75_6F 61 07|00_64|FA 30 31 74 t10....aout100.. 002B81
 31 2B 66_6E 69 07|00_64|35 32 2B 30 74_75 6F 07 .out0+25d..inf+1 002B91
 6F 07|00_2C|35 35 35 2D 35 6E 69 07|02_2B|30 30 00+..in5-555,..o 002BA1
 00 00 00 00 00 00 00 00 00 00 34 34 2D 66 74 75 utf-44.......... 002BB1
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 002BC1
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 002BD1
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 002BE1
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 002BF1
```

↓

```
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005201
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005211
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005221
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005231
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005241
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005251
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005261
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005271
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005281
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005291
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 0052A1
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 0052B1
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 0052C1
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 0052D1
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 0052E1
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 0052F1
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005301
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................ 005311
```

Example Name Text Block:

This is the first Name text block, designated block 0. In this example, block 0 is the Name text block for inputs I0017 through I0032. Only input I0020 has Name text. Each reference with Name text occupies 21 characters within the block.

```
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005321
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005331
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005341
|61|20_20_20_20_20_20_20_20_20_20_20_20_20_20_20_20                a 005351
|61_6E_20_20_20_20_30_32_20_20_20_74 75 70 6E 69|input   20     na 005361
20 20 20 20 20 20 20 20 20 20 20 20|20_20_65_6D|me                 005371
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005381
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005391
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0053A1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0053B1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0053C1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0053D1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0053E1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0053F1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005401
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005411
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005421
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005431
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005441
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005451
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005461
```

Example Name Text Block: (free block)

This is block 1. In this example, it is the first free text block. A free text block is one that formerly contained text, but is now empty. A pointer in the header of the .NAM file contains the number of the first free text block. The first two bytes of the block contain a pointer to the next free text block (07).

```
20 20 20 20 20 20 20 20 20 20 20 20 20 20|00_07|                  005471
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005481
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005491
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0054A1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0054B1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0054C1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0054D1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0054E1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0054F1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005501
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005511
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005521
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005531
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005541
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005551
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005561
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005571
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005581
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005591
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0055A1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0055B1
```

Example Name Text Block:

This is block 2. In the example, it is the Name text block for inputs I0001 through I0016. Only input I0006 has Name text.

```
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0055C1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0055D1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0055E1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0055F1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005601
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005611
|20 74 75 70 6E 69|20 20 20 20 20 20 20 20 20 20|           input  005621
20|20 20 20 65 6D 61 6E 20 20 20 20 36 20 20 20|    6     name     005631
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005641
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005651
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005661
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005671
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005681
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005691
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0056A1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0056B1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0056C1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0056D1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0056E1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0056F1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005701
```

Example Name Text Block:

Block 3 is the Name text block for outputs O0017 through O0032. Only output O0020 has Name text.

```
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005711
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005721
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005731
|6F|20 20 20 20 20 20 20 20 20 20 20 20 20 20 20|               o 005741
|61 6E 20 20 20 20 30 32 20 20 20 74 75 70 74 75|utput   20    na 005751
20 20 20 20 20 20 20 20 20 20 20 20|20 20 65 6D|me              005761
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005771
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005781
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005791
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0057A1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0057B1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0057C1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0057D1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0057E1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0057F1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005801
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005811
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005821
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005831
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005841
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  005851
```

Example Name Text Block:

Block 4 is the Name text block for registers R15983 through R16000. Only R16000 has Name text.

```
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005861
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005871
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005881
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005891
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0058A1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0058B1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0058C1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0058D1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0058E1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0058F1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005901
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005911
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005921
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005931
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005941
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005951
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005961
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005971
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005981
|67 65 72 20 20|20 20 20 20 20 20 20 20 20 20                reg   005991
|20 20 65 6D 61 6E 20 20 30 30 30 36 31 20 20 20|   16000  name    0059A1
```

Example Name Text Block:

Block 5 is the Name text block for outputs OE-0993 through OE-1008. In this example, OE-1000 has Name text.

```
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0059B1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0059C1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0059D1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0059E1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    0059F1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005A01
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005A11
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005A21
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005A31
|6E 72 65 74 6E 69 20 65 20 74 75 6F 20|20 20 20     out e intern 005A41
 20 20 20 20 20 20 20 20|20 20 30 30 30 31 20 20|  1000             005A51
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                   005A61
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                   005A71
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                   005A81
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                   005A91
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                   005AA1
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                   005AB1
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                   005AC1
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                   005AD1
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                   005AE1
 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                   005AF1
```

Example Name Text Block:

Block 6 is the Name text block for inputs I2+0033 through I2+0048. Input I2+0047 has Name text.

```
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005B01
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005B11
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005B21
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005B31
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005B41
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005B51
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005B61
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005B71
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005B81
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005B91
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005BA1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005BB1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005BC1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005BD1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005BE1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005BF1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005C01
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005C11
|74 78 65 32 20 74 75 70 6E 69 20|20 20 20 20 20        input 2ext 005C21
20 20 20 20 20|20 20 20 37 34 20 20 20 6E 72 65|ern    47          005C31
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005C41
```

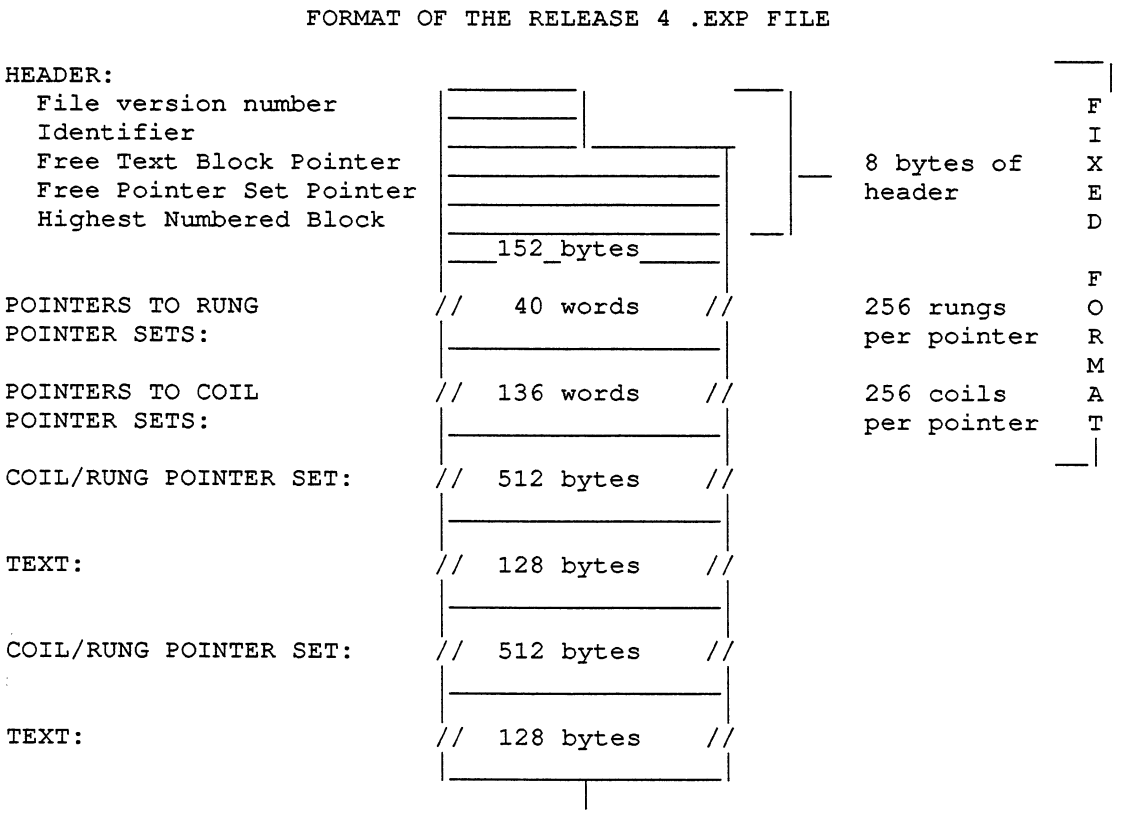
Example Name Text Block: (free block)

In the example file, block 7 is the last free text block. This is shown by the first two bytes, which contain the nil pointer FFFF.

```
20 20 20 20 20 20 20 20 20 20 20 20 20 20|FF FF|                  005C51
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005C61
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005C71
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005C81
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005C91
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005CA1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005CB1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005CC1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005CD1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005CE1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005CF1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005D01
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005D11
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005D21
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005D31
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005D41
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005D51
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005D61
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005D71
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005D81
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    005D91
```

.EXP File

The .EXP file structure is based on blocks of 128 bytes. The file consists of a file header of 512 bytes. This portion of the file has a fixed format. The rest of the file consists of pointer sets and text blocks. These can be in any order.



.EXP File Header

The .EXP file header (512 bytes), contains file description information, 40 Pointers to Rung Pointer sets, 136 Pointers to Coil Pointer sets and 152 unused bytes. Header contents are described below:

1. File version number (1 byte)

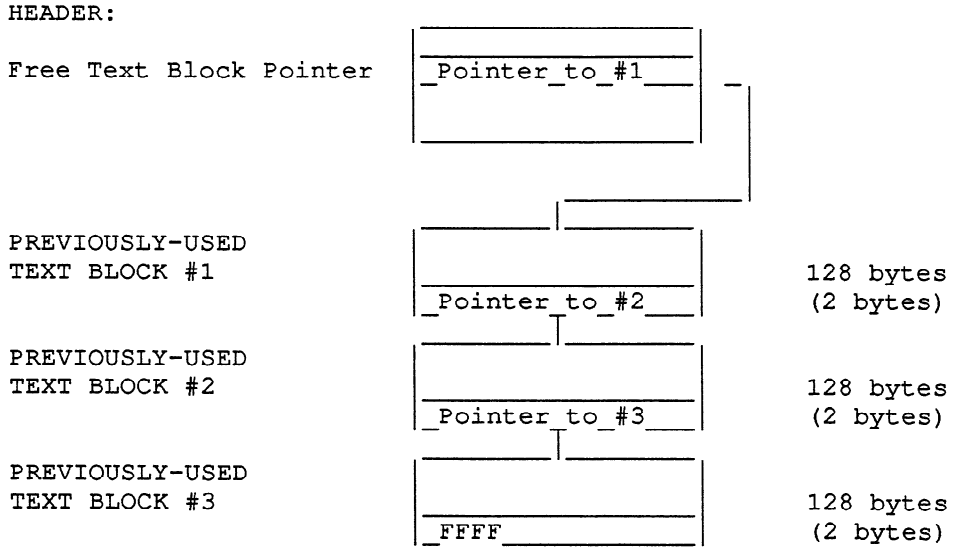
   The file version number is:

LM6 Release	Version
1.02	1
1.03	2
2.01	3
3.01	2
4.01	2

2. Unused byte (1 byte)
3. Free text-block pointer (1 word)

   The free text-block pointer references the first block in a linked list of previously-used but now-empty text blocks for coil labels and rung explanations. If no previously-used text block exists, the free text-block pointer is a nil pointer.

   Each text block contains 128 bytes. The last two bytes are a pointer to the next block in the list. For example, the block number of the second free-text block is in the last two bytes of the first free-text block. Subsequent blocks are linked in the same way, as shown below. The last block in the linked list has a nil pointer (value of FFFF).

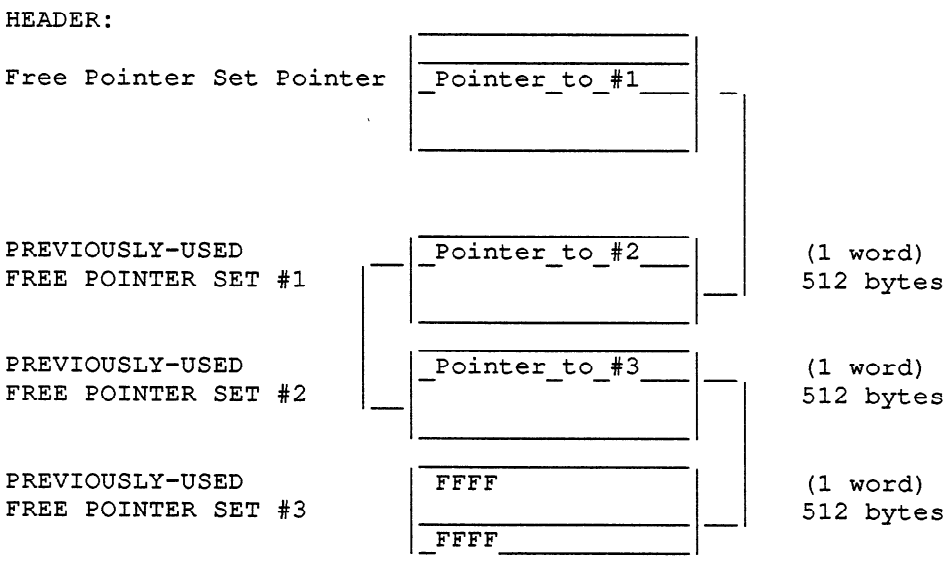


4. Free pointer-set pointer (1 word)

   The free pointer-set pointer references the first pointer set in a linked list of previously-used but now-empty pointer sets for coil labels and rung explanations.

   Each pointer set consists of four consecutive 128-byte blocks. The first word of each set is a block-number pointer to the next set, as illustrated below. For example, the block number of the second free pointer set is in the first word of the first free pointer set. The block number of the third free pointer set is in the first word in the second free pointer set. The last free pointer set in the list has a nil pointer (FFFF) in its first word.

   If no previously-used pointer sets exist, the free pointer-set pointer is a nil pointer.



5. Highest block number (1 word)

   The block number of the last block in the .EXP file. Each block has 128 bytes. The first block number is block 0, which follows the file header. The highest block number is one less than the number of 128-byte blocks after the file header. For example, in a file with one pointer set and four text blocks, the highest block number is 7:

   a. The pointer set occupies four consecutive 128-byte blocks.

   b. The four text blocks occupy 128 bytes.

   c. Therefore, there are eight 128-byte .EXP file blocks.

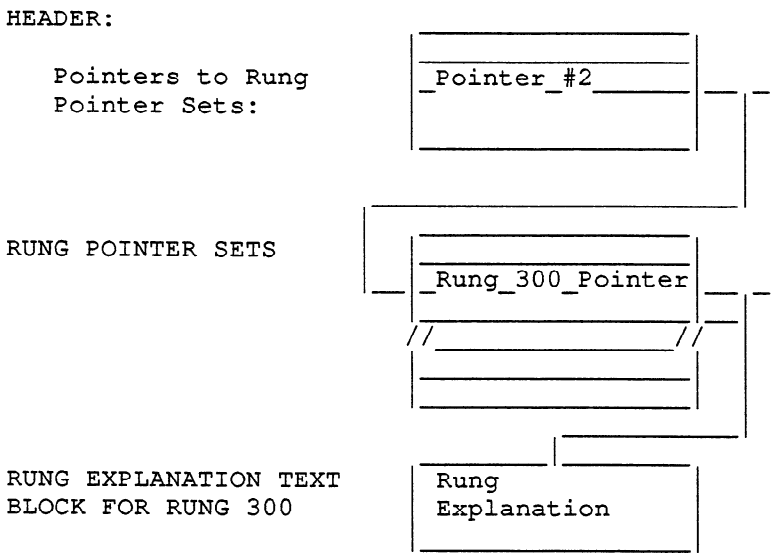
   d. Since the block number of the first block is 0, the highest block number in this .EXP file is 7.

   The highest block number never decreases. It increments by 4 as new pointer sets and new text blocks (in groups of 4 consecutive text blocks) are added to the .EXP file.

6. Unused
7. Pointers to Rung Pointer Sets (40 words)

   The next 80 bytes of the .EXP file header consists of the Pointers to Rung Pointers sets. There are 40 one-word pointers in this portion of the file. Each one consists of the file block number of a Rung Pointer set. Individual pointers within those sets access the Rung Explanation text.

   For example, if rung 300 has explanation text, the second Pointer to Rung Pointer set entry in the file header contains the block number of the Rung Pointer set for rungs 256-511. In that set of 256 rung pointers, the thirteenth word contains the block number of the first block of Rung Explanation text for rung 300.



A nil pointer (FFFF) in a Pointer to Rung Pointer set means none of the 256 rungs associated with that set have Rung Explanation text.

When a new Rung Pointer set is needed and the Free Pointer set pointer is a nil pointer (meaning that there are no available previously-created empty sets), a new Rung Pointer set is created at the end of the .EXP file.

8. Pointers to Coil-Pointer Sets (136 words)

The next 272 bytes of the header consists of the Pointers to Coil Pointer sets. There are 136 one-word pointers. Each pointer accesses a Coil Pointer set that represents 256 output addresses. Therefore, four entries in the Pointers to Coil Pointer sets in the header can access a total of 1024 output addresses. There are four types of Pointers to Coil Pointer sets, one for each output coil type, in this sequence:

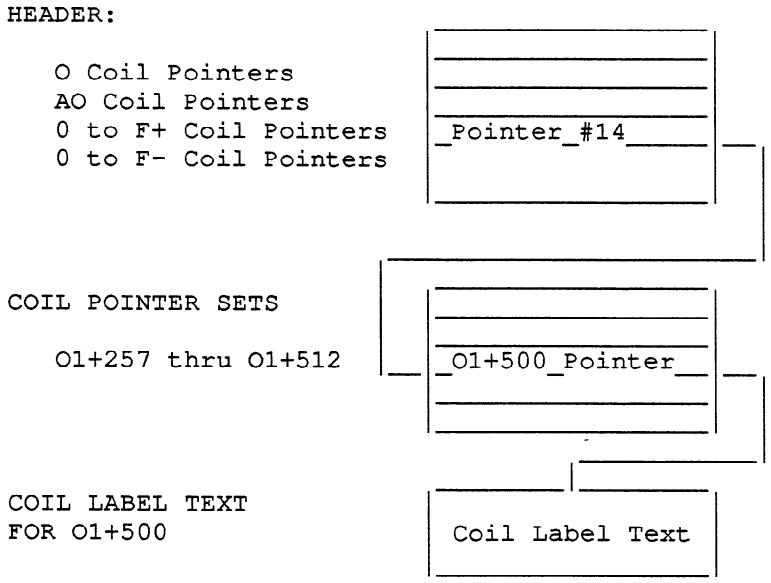
O (output): four entries = four pointers

AO (auxiliary output): four entries = four pointers

External Channels 0-f: sixteen channels x four entries = 64 pointers

Internal Channels 0-f: sixteen entries x four entries = 64 pointers

Each pointer consists of the block number of a Coil Pointer set. Individual pointers within the Coil Pointer sets access the Coil Label text blocks. For example, if O1+500 has Coil Label text, the fourteenth pointer in the Pointers to Coil Pointers set in the header contains the beginning block number of the Coil Label pointers for O1+257 through O1+512. In that set of 256 Coil Label pointers, the 244th word contains the number of the first block of Coil Label text for O1+500.



A nil pointer (FFFF) in a Pointer to Coil Label Set means none of the 256 output coils associated with that set have Coil Label text.

Within a set, if an individual Coil Label pointer is nil (value of FFFF), no Coil Label text exists for that output coil. If a Coil Label Pointer set consists of all nil pointers, that set is placed on the Free Pointer Set list. When a new Coil Label Pointer set is needed and the Free Pointer Set Pointer is a nil pointer, a new Coil Label Pointer Set is created at the end of the .EXP file.

9. 152 unused bytes

The end of the .EXP file header consists of 152 unused bytes.

Rung Pointer Sets

A rung pointer contains the beginning block number of the Rung Explanation text for a rung. If a rung pointer is nil (FFFF), there is no Rung Explanation text for that rung.

Each Rung Pointer set occupies four contiguous file blocks, and stores 256 rung pointers. Individual rung pointers within the set are accessed by the Pointers to Rung Pointer sets in the header, as explained previously.

If the entire Rung Pointer set consists of nil pointers, that Rung Pointer set is placed on the Free Pointer set list. Pointer sets on the Free Pointer set list are accessed by the Free Pointer set pointer in the header, as explained previously.

Coil Pointer Sets

A coil pointer contains the beginning block number of the Coil Label text for an output coil. If a coil pointer is nil (FFFF), there is no Coil Label text for that output.

Each Coil Pointer set occupies 4 contiguous file blocks which together contain 256 coil pointers. Individual coil pointers within the set are accessed by the Pointers to Coil Pointer sets in the header, as explained previously.

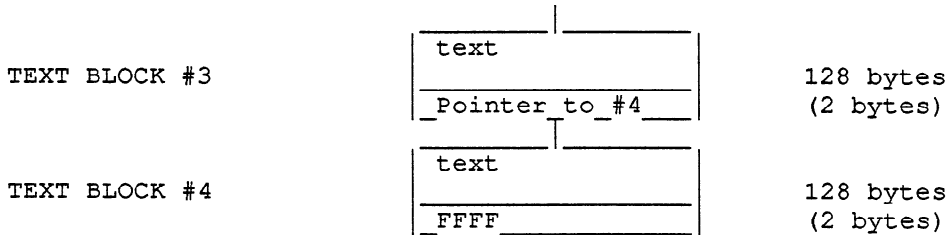
If the entire Coil Pointer set consists of nil pointers, that Coil Pointer set is placed on the Free Pointer set list. Pointer sets on the Free Pointer set list are accessed by the Free Pointer set pointer in the header, as explained previously.

Text Blocks

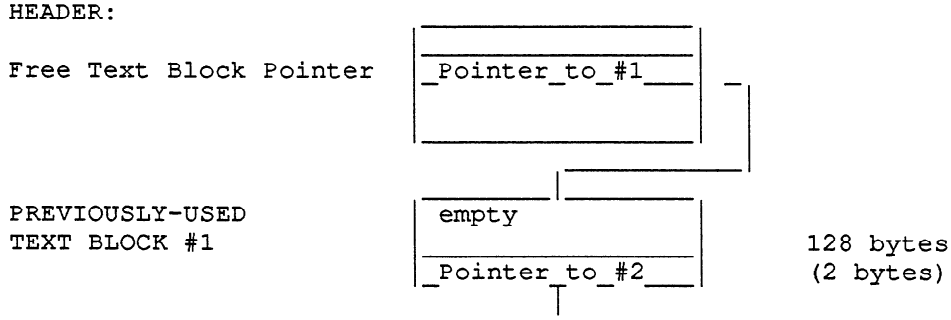
A text block contains any of the standard ASCII keyboard characters. The text should be viewed as a buffer of contiguous characters, including carriage return characters (1E hexadecimal). The last text character is an End-of-Buffer character (17 hexadecimal).

Structure of Text Blocks

Text blocks for coil labels and rung explanations are identical in structure. Each text block is 128 bytes. The first 126 bytes contain text. The last 2 bytes contain the block number of the next block of text for that label or explanation. Text blocks for a label or explanation are linked throughout the .EXP file. The last text block in the linked list of blocks has a nil pointer (FFFF) in its last 2 bytes.



If the number of characters in a label or explanation is reduced, the number of text blocks necessary to contain the text may decrease. If that happens, the unused text blocks are linked to each other, like text blocks, with file block numbers for the next free block in the last 2 bytes. This list of free blocks is linked into the Free text block list maintained with the Free text block pointer in the .EXP file header.



If the label or explanation is cleared by blanking it, all text blocks are added to the list of free blocks, and the pointer in the pointer set which pointed to the first text block will be the nil pointer (FFFF).

As text block requirements fluctuate with label and explanation sizes, text blocks are taken from and added to the linked list of Free text blocks. If no available (previously-created, now-empty) text blocks exist when a new text block is required, four text blocks are created at the end of the .EXP file. After storing text in as many new text blocks as needed, the remaining text blocks of the four new text blocks are added to the Free text block list.

.EXP File Structure

The following pages show the structure of the .EXP file.

.EXP File Header

The content of the .EXP file header has a fixed format:

Byte Number	
1	version_number
2	unused
3, 4	free_text-block_pointer
5, 6	free_pointer-block_pointer
7, 8	highest_block_number
9-160	unused
161-240	pointers_to_Rung_Pointer_Sets
241-248	ptrs_to_Coil_Ptr_Sets_O_1-1024
249-256	ptrs_to_Coil_Ptr_Sets_AO_1-1024
257-264	ptrs_to_Coil_Ptr_Sets_O0+_1-1024
265-272	ptrs_to_Coil_Ptr_Sets_O1+_1-1024
//	//
377-384	ptrs_to_Coil_Ptr_Sets_Of+_1-1024
385-392	ptrs_to_Coil_Ptr_Sets_O0-_1-1024
//	//
497-504	ptrs_to_Coil_Ptr_Sets_Oe-_1-1024
505-512	ptrs_to_Coil_Ptr_Sets_Of-_1-1024

The Rung and Coil Pointer sets and the text blocks occupy the remainder of the .EXP file. The format of these parts of the file is shown on the next page.

Pointers and Text Blocks

After the header, the .EXP file contains the Pointer sets and text blocks. These are created in the file as needed, and can be in any order. The basic format for each of these is shown below.

Rung Pointer Set (4 consecutive file blocks)

Each Rung Pointer set has the format shown below. This is repeated for each Rung Pointer set required. Note: N is unique for each Rung Pointer set, and may be one of the following values: 0, 256, 512, ... 9984.

Byte Number	
1-128	rung_N_thru_rung_N+63
129-256	rung_N+64_thru_rung_N+127
257-384	rung_N+128_thru_rung_N+191
385-512	rung_N+192_thru_rung_N+255

Coil Pointer Set (4 consecutive file blocks)

Each Coil Pointer set has the format shown below. This is repeated for each Coil Pointer set required. Note: N assumes the following values for each output reference type: 1, 257, 513, 769.

Byte Number	
1-128	coil_address_N_thru_N+63
129-256	coil_address_N+64_thru_N+127
257-384	coil_address_N+128_thru_N+191
385-512	coil_address_N+192_thru_N+255

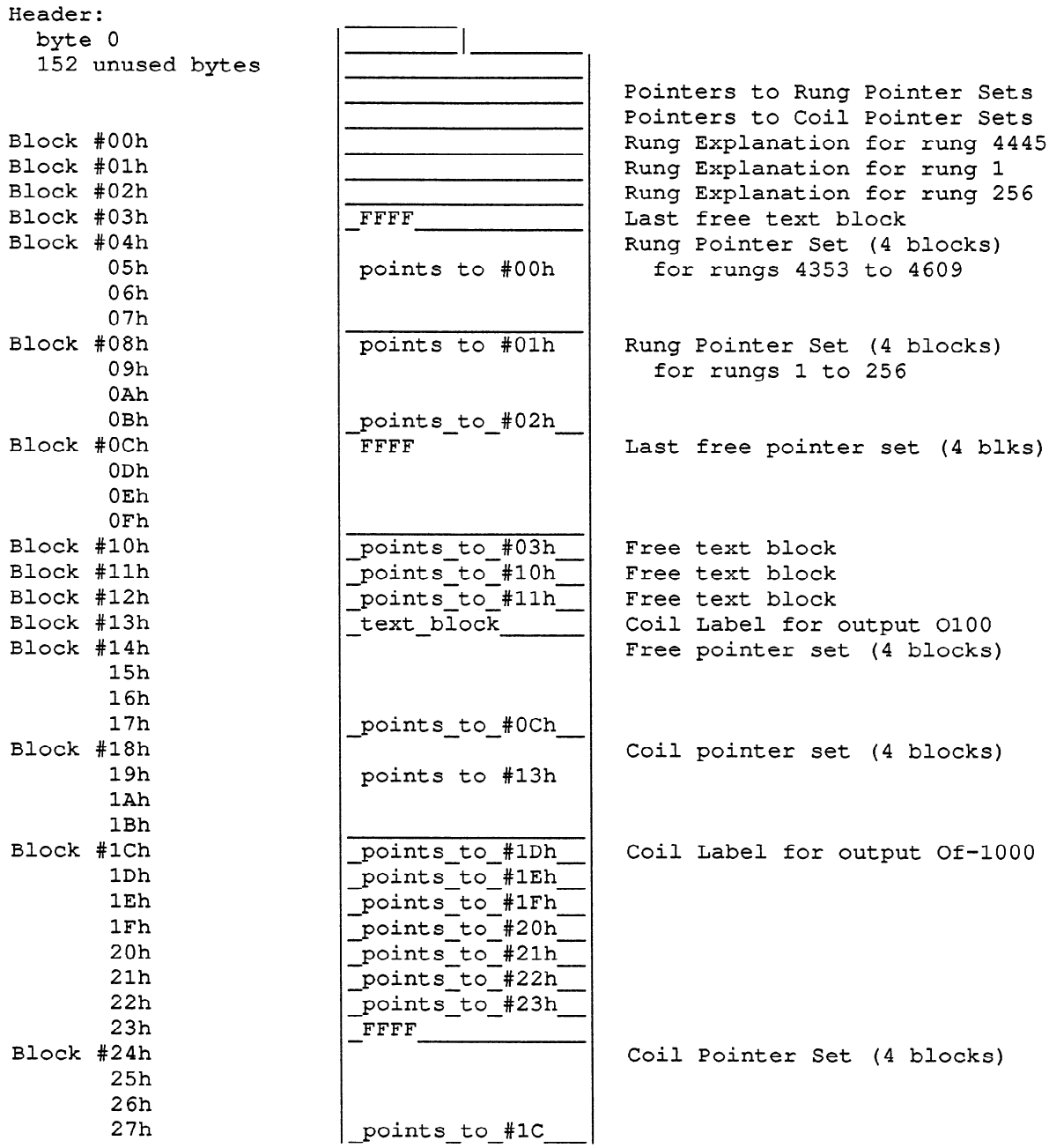
Text Blocks

Each text block for rung explanations or coil labels has the following format. This is repeated for additional text, as required.

Byte Number	
1-126	text
127-128	pointer_to_another_text_block

An annotated example .EXP file for release 4 Logicmaster 6 software appears on the following pages. The illustration below shows the basic arrangement of data in the example file.

FORMAT OF THE EXAMPLE .EXP FILE



NOTE

In the example file contents that follow, data and block numbers are in hexadecimal.

Example .EXP File Header

First 8 bytes:

```
00 27 00 14 00 12 2A 02 < first byte, byte 0
                  |__ version number
               |_____ identifier
            |________ free text block pointer
         |___________ free pointer block pointer
      |______________ highest numbered block
```

152 unused bytes:

```
FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Pointers to Rung Pointer Sets

In the example file, there are two Pointers to Rung Pointer sets (shown here in boxes). The first is a pointer to the Rung Pointer set beginning at block 08h. The second is a pointer to the Rung Pointer set beginning at block 04h.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF|00 08|< first entry
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF|00 04|FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Pointers to Coil Pointer Sets

In this file, there are two Pointers to Coil Pointer sets. The first is a pointer to the Coil Pointer set beginning at block 18h. The second is a pointer to the Coil Pointer set beginning at block 24h.

```
        Aux Output                          Output
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF|00_18|< first entry
              O1+                                 O0+
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              O3+                                 O2+
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              O5+                                 O4+
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              O7+                                 O6+
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              O9+                                 O8+
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              Ob+                                 Oa+


FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              Od+                                 Oc+
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              Of+                                 Oe+
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              O1-                                 O0-
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              O3-                                 O2-
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              O5-                                 O4-
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              O7-                                 O6-
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              O9-                                 O8-
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              Ob-                                 Oa-
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              Od-                                 Oc-
FF FF FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
              Of-                                 Oe-
|00_24|FF FF FF FF FF FF             FF FF FF FF FF FF FF FF
```

Block # Oh

This is a text block. It contains the Rung Explanation text for rung **4445.** Each byte in the block represents one ASCII character. The last character of the rung explanation is the End-of-Buffer character (17 hex). The hex value 20 represents a blank character. The last two bytes in the block contain the nil pointer FFFF, indicating that no block is linked to the end of this one.

																File Byte # (hex)
20	20	20	20	20	17	1E	2E	34	34	34	34	20	50	58	45	EXP 4444. 000200
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000210
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000220
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000230
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000240
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000250
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000260
FF	FF	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000270

Block # lh

This text block contains the Rung Explanation text for rung 1. It has the same format as the previous block. This block also ends with the nil pointer FFFF.

20	20	20	20	20	20	20	20	17	1E	2E	30	20	50	58	45	EXP 0. 000280
20	20	20	20	20	20	20	20	20	-20	20	20	20	20	20	20	000290
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	0002A0
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	0002B0
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	0002C0
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	0002D0
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	0002E0
FF	FF	20	20	20	20	20	20	20	20	20	20	20	20	20	20	0002F0

Block # 2h

This block contains the Rung Explanation text for rung 256. It ends with the nil pointer FFFF.

20	20	20	20	20	20	17	1E	2E	35	35	32	20	50	58	45	EXP 255. 000300
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000310
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000320
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000330
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000340
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000350
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000360
FF	FF	20	20	20	20	20	20	20	20	20	20	20	20	20	20	000370

Block # 3h (free text block)

This is a previously-used text block which is now empty. It ends with the nil pointer FFFF.

```
20 20 20 20 20 20 20 20 20 20 20 17 1E 5A 5A 5A ZZZ              000380
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000390
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0003A0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0003B0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0003C0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0003D0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000330
FF FF 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0003F0
```

Block # 4h-7h (rung pointer set)

In the example file, blocks 04h through 07h contain the Rung Pointer set for rungs 4353 through 4609. There is one Rung Pointer used. It contains the beginning block number (00h) for the Rung Explanation text for rung 4445.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000400
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000410
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000420
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000430
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000440
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000450
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000460
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000470
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000480
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000490
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0004A0
FF FF FF FF FF FF|00 00|FF FF FF FF FF FF FF FF  ................  0004B0
FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF FF  ................  0004C0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0004D0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0004E0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0004F0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000500
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000510
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000520
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000530
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000540
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000550
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000560
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000570


FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000580
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000590
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0005A0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0005B0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0005C0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0005D0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0005E0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0005F0
```

Block # 8h-Bh (rung pointer set)

Blocks 08h through 0Bh contain the Rung Pointer set for rungs 1 through 256. It contains the beginning block number (Olh) for the Rung Explanation text for rung 1, and the block number (02h) for the text for rung 256.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF|0001|.............. 000600
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF-FF ..............  000610
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000620
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000630
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000640
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000650
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000660
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000670
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000680
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000690
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0006A0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0006B0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0006C0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0006D0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0006E0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0006F0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000700
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000710
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000720
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000730
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000740
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000750
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000760
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000770
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000780
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  000790
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0007A0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0007B0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0007C0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0007D0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0007E0
|00 02|FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..............  0007F0
```

Block # Ch-Fh (free pointer set)

Blocks 0Ch through 0Fh contain a previously-used but now-empty pointer set. The first two bytes are the nil pointer FFFF, because this is the last free pointer set.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF|FF|FF  ................  000800
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF-FF  ................  000810
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000820
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000830
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000840
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000850
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000860
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000870
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000880
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000890
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0008A0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0008B0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0008C0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0008D0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0008E0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0008F0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000900
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000910
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000920
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000930
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000940
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000950
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000960
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000970
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000980
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  000990
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0009A0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0009B0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0009C0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0009D0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0009E0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................  0009F0
```

Block # 10h (free text block)

Block 10h is a free text block. The last two bytes are a pointer to the next free text block (03h).

```
20 20 20 20 20 20 20 20 20 20 20 20 17 5A 5A 5A ZZZ              000A00
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000A10
20  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                 000A20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000A30
20  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                 000A40
20  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                 000A50
20  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                 000A60
|00 03|20 - 20 20 20 20 20 20 20 20 20 20 20 20 20 20           000A70
```

Block # llh (free text block)

Block 1 lh is a free text block. The last two bytes are a pointer to the next free text block (10h).

```
20 20 20 20 20 20 20 20 20 20 20 20 17 5A 5A 5A ZZZ              000A80
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000A90
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000AA0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000AB0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000AC0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000AD0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000AE0
|00_10|20 - 20 20 20 20 20 20 20 20 20 20 20 20 20 20            000AF0
```

Block # 12h (free text block)

Block 12h is a free text block. The last two bytes are a pointer to the next free text block (1 lh).

```
20 20 20 20 20 20 20 20 20 20 20 20 17 5A 5A 5A ZZZ              000B00
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000B10
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000B20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000B30
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000B40
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000B50
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000B60
|00_11|20 - 20 20 20 20 20 20 20 20 20 20 20 20 20 20            000B70
```

Block # 13h

Block 13h is a text block. It contains the text of the coil label for output 0100. The last two bytes are a nil pointer indicating that no additional text blocks are linked to the end of this one.

```
20 4C 49 4F 43 20 30 30 31 20 54 55 50 54 55 4F OUTPUT 100 COIL  000B80
20 20 20 20 20 20 20 20 20 17 1E 4C 45 42 41 4C LABEL            000B90
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000BA0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000BB0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000BC0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000BD0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  000BE0
|FF - FF|20 20 20 20 20 20 20 20 20 20 20 20 20 20               000BF0
```

Block # 14h-17h (free pointer set)

Blocks 14h through 17h (continued here on the next page) contain a previously-used but now empty pointer set. The first two bytes are a pointer to the next free pointer set (0Ch).

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF|00 0C| ..............  000C00
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF-FF  ..............  000C10
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000C20
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000C30
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000C40
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000C50
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000C60
                                                        continued . . .
```

continued . . .

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000C70
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000C80
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000C90
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000CA0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000CB0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000CC0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000CD0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000CE0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000CF0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000D00
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000D10
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000D20
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000D30
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000D40
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000D50
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000D60
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000D70
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000D80
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000D90
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000DA0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000DB0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000DC0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000DD0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000DE0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ..............  000DF0
```

Block # 18h-1Bh (coil pointer set)

Blocks 18h through 1Bh contain the Coil Pointer set fur output coils 0 through 255. It contains the beginning block number (13h) for the coil label for coil 0100.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E00
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E10
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E20
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000330
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E40
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E50
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E60
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E70
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E80
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000E90
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000EA0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000EB0
FF FF FF FF FF FF FF FF|00 13|FF FF FF FF FF FF ................ 000EC0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000ED0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000EE0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000EF0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000F00
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000F10
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000F20
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000F30
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000F40
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000F50
```

continued . . .

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000F60
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000F70
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000F80
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000F90
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000FA0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000FB0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000FC0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000FD0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000FE0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 000FF0
```

Block # 1Ch

Block 1Ch is a text block. It contains the beginning of the Coil Label text for output Of-1000. The last two bytes are a pointer to block 1D, which continues the Coil Label text for this output.

```
46 20 4C 45 4E 4E 41 48 43 20 54 55 50 54 55 4F OUTPUT CHANNEL F oolooo
43 20 30 30 30 31 20 4C 41 4E 52 45 54 4E 49 20  INTERNAL 1000 C 001010
54 55 50 54 55 4F 1E 4C 45 42 41 4C 20 4C 49 4F OIL LABEL OUTPUT 001020
52 45 54 4E 49 20 46 20 4C 45 4E 4E 41 48 43 20  CHANNEL F INTER 001030
41 4C 20 4C 49 4F 43 20 30 30 30 31 20 4C 41 4E NAL 1000 COIL LA 001040
4E 4E 41 48 43 20 54 55 50 54 55 4F 1E 4C 45 42 BEL OUTPUT CHANN 001050
30 31 20 4C 41 4E 52 45 54 4E 49 20 46 20 4C 45 EL F INTERNAL 10 001060
00 1D 1E _ 4C 45 42 41 4C 20 4C 49 4F 43 20 30 30 00 COIL LABEL    001070
```

Block # 1Dh

Blocks 1Dh through 23h are text blocks. They contain the rest of the Coil Label text for output Of-1000. The last two bytes of each block link it to the next block. The last two bytes of block 23h contain the nil pointer FFFF.

```
46 20 4C 45 4E 4E 41 48 43 20 54 55 50 54 55 4F OUTPUT CHANNEL F 001080
43 20 30 30 30 31 20 4C 41 4E 52 45 54 4E 49 20  INTERNAL 1000 C 001090
54 55 50 54 55 4F 1E 4C 45 42 41 4C 20 4C 49 4F OIL LABEL OUTPUT 0010A0
52 45 54 4E 49 20 46 20 4C 45 4E 4E 41 48 43 20  CHANNEL F INTER 0010B0
41 4C 20 4C 49 4F 43 20 30 30 30 31 20 4C 41 4E NAL 1000 COIL LA 0010C0
4E 4E 41 48 43 20 54 55 50 54 55 4F 1E 4C 45 42 BEL OUTPUT CHANN 0010D0
30 31 20 4C 41 4E 52 45 54 4E 49 20 46 20 4C 45 EL F INTERNAL 10 0010E0
00 1E 1E _ 4C 45 42 41 4C 20 4C 49 4F 43 20 30 30 00 COIL LABEL    0010F0
```

Block # 1Eh

```
46 20 4C 45 4E 4E 41 48 43 20 54 55 50 54 55 4F OUTPUT CHANNEL F 001100
43 20 30 30 30 31 20 4C 41 4E 52 45 54 4E 49 20  INTERNAL 1000 C 001110
54 55 50 54 55 4F 1E 4C 45 42 41 4C 20 4C 49 4F OIL LABEL OUTPUT 001120
52 45 54 4E 49 20 46 20 4C 45 4E 4E 41 48 43 20  CHANNEL F INTER 001130
41 4C 20 4C 49 4F 43 20 30 30 30 31 20 4C 41 4E NAL 1000 COIL LA 001140
4E 4E 41 48 43 20 54 55 50 54 55 4F 1E 4C 45 42 BEL OUTPUT CHANN 001150
30 31 20 4C 41 4E 52 45 54 4E 49 20 46 20 4C 45 EL F INTERNAL 10 001160
00 1F 1E _ 4C 45 42 41 4C 20 4C 49 4F 43 20 30 30 00 COIL LABEL    001170
```

Block # 1Fh

```
46 20 4C 45 4E 4E 41 48 43 20 54 55 50 54 55 4F OUTPUT CHANNEL F 001180
43 20 30 30 30 31 20 4C 41 4E 52 45 54 4E 49 20  INTERNAL 1000 C 001190
54 55 50 54 55 4F 1E 4C 45 42 41 4C 20 4C 49 4F OIL LABEL OUTPUT 0011A0
52 45 54 4E 49 20 46 20 4C 45 4E 4E 41 48 43 20  CHANNEL F INTER 0011B0
41 4C 20 4C 49 4F 43 20 30 30 30 31 20 4C 41 4E NAL 1000 COIL LA 0011C0
4E 4E 41 48 43 20 54 55 50 54 55 4F 1E 4C 45 42 BEL OUTPUT CHANN 0011D0
30 31 20 4C 41 4E 52 45 54 4E 49 20 46 20 4C 45 EL F INTERNAL 10 0011E0
|00 20|1E _ 4C 45 42 41 4C 20 4C 49 4F 43 20 30 30 00 COIL LABEL  0011F0
```

Block # 20h

```
46 20 4C 45 4E 4E 41 48 43 20 54 55 50 54 55 4F OUTPUT CHANNEL F 001200
43 20 30 30 30 31 20 4C 41 4E 52 45 54 4E 49 20  INTERNAL 1000 C 001210
54 55 50 54 55 4F 1E 4C 45 42 41 4C 20 4C 49 4F OIL LABEL OUTPUT 001220
52 45 54 4E 49 20 46 20 4C 45 4E 4E 41 48 43 20  CHANNEL F INTER 001230
41 4C 20 4C 49 4F 43 20 30 30 30 31 20 4C 41 4E NAL 1000 COIL LA 001240
4E 4E 41 48 43 20 54 55 50 54 55 4F 1E 4C 45 42 BEL OUTPUT CHANN 001250
30 31 20 4C 41 4E 52 45 54 4E 49 20 46 20 4C 45 EL F INTERNAL 10 001260
|00 21|1E _ 4C 45 42 41 4C 20 4C 49 4F 43 20 30 30 00 COIL LABEL  001270
```

Block # 21h

```
46 20 4C 45 4E 4E 41 48 43 20 54 55 50 54 55 4F OUTPUT CHANNEL F 001280
43 20 30 30 30 31 20 4C 41 4E 52 45 54 4E 49 20  INTERNAL 1000 C 001290
54 55 50 54 55 4F 1E 4C 45 42 41 4C 20 4C 49 4F OIL LABEL OUTPUT 0012A0
52 45 54 4E 49 20 46 20 4C 45 4E 4E 41 48 43 20  CHANNEL F INTER 0012B0
41 4C 20 4C 49 4F 43 20 30 30 30 31 20 4C 41 4E NAL 1000 COIL LA 0012C0
4E 4E 41 48 43 20 54 55 50 54 55 4F 1E 4C 45 42 BEL OUTPUT CHANN 0012D0
30 31 20 4C 41 4E 52 45 54 4E 49 20 46 20 4C 45 EL F INTERNAL 10 0012E0
|00 22|1E _ 4C 45 42 41 4C 20 4C 49 4F 43 20 30 30 00 COIL LABEL  0012F0
```

Block # 22h

```
46 20 4C 45 4E 4E 41 48 43 20 54 55 50 54 55 4F OUTPUT CHANNEL F 001300
43 20 30 30 30 31 20 4C 41 4E 52 45 54 4E 49 20  INTERNAL 1000 C 001310
54 55 50 54 55 4F 1E 4C 45 42 41 4C 20 4C 49 4F OIL LABEL OUTPUT 001320
52 45 54 4E 49 20 46 20 4C 45 4E 4E 41 48 43 20  CHANNEL F INTER 001330
41 4C 20 4C 49 4F 43 20 30 30 30 31 20 4C 41 4E NAL 1000 COIL LA 001340
4E 4E 41 48 43 20 54 55 50 54 55 4F 1E 4C 45 42 BEL OUTPUT CHANN 001350
30 31 20 4C 41 4E 52 45 54 4E 49 20 46 20 4C 45 EL F INTERNAL 10 001360
|00 23|1E _ 4C 45 42 41 4C 20 4C 49 4F 43 20 30 30 00 COIL LABEL  001370
```

Block # 23h

```
46 20 4C 45 4E 4E 41 48 43 20 54 55 50 54 55 4F OUTPUT CHANNEL F 001380
43 20 30 30 30 31 20 4C 41 4E 52 45 54 4E 49 20  INTERNAL 1000 C 001390
54 55 50 54 55 4F 1E 4C 45 42 41 4C 20 4C 49 4F OIL LABEL OUTPUT 0013A0
52 45 54 4E 49 20 46 20 4C 45 4E 4E 41 48 43 20  CHANNEL F INTER 0013B0
41 4C 20 4C 49 4F 43 20 30 30 30 31 20 4C 41 4E NAL 1000 COIL LA 0013C0
20 20 20 20 20 20 20 20 20 20 17 1E 4C 45 42 20 BEL              0013D0
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0013E0
FF FF 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  0013F0
```

Block # 24h-27h (coil pointer set)

Block 24h is a Coil Pointer set. It contains the beginning block address (1Ch) of the Coil Label text for output Of- 1000.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001400
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001410
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001420
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001430
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001440
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001450
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001460
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001470
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001480
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001490
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0014A0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0014B0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0014C0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0014D0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0014E0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0014F0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001500
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001510
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001520
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001530
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001540
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001550
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001560
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001570
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001580
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 001590
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0015A0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0015B0
00 1C FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0015C0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0015D0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0015E0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................ 0015F0
```

A

B

C

E

F

G

H

I

K

L

M

N

0

P

R

S

T

U

V

W

GE Fanuc Automation North America, Inc., Charlottesville, Virginia

CUT ON DOTTED LINE

READER'S COMMENTS

GFZ-005OA

We invite your comments and welcome suggestions to make this manual more useful.

Publication No. _____ Date of Publication _____ Today's Date _____

GENERAL COMMENTS:

	Improve	Acceptable	Good	Excellent
Contents	☐	☐	☐	☐
Organization	☐	☐	☐	☐
Accuracy	☐	☐	☐	☐
Clarity	☐	☐	☐	☐
Completeness	☐	☐	☐	☐
Examples/Illustrations	☐	☐	☐	☐
Referencing/Indexing	☐	☐	☐	☐
Readability	☐	☐	☐	☐

DETAILED COMMENTS: (Correct, expand, etc. – Please be specific.)

Page No. **Comment**

_____ _____

_____ _____

_____ _____

_____ _____

Other suggestions for improving this document: _____

_____

As compared to other manufacturers of a similar product, how would you rate this document ?

Superior ☐ Comparable ☐ Inferior ☐ Don't Know ☐

Commments: _____

_____

Are you interested in subscribing to a documentation update plan? Yes ☐ No ☐

APPLICATION INFORMATION:

Indicate the type of user/reader function that you most nearly represent:

☐ System Designer ☐ Programmer
☐ Distributor ☐ Maintenance
☐ OEM ☐ Operator
☐ Installation ☐ Other (Please Specify) _____

Type of Equipment: ☐ Series 90-70 ☐ Series 90-30 ☐ Series 90-20 •1 Series Six
☐ Series Five ☐ Series One ☐ Genius I/O ☐ Other

Comments concerning your specific application: _____

_____

_____

_____

From:

Name: _____

Title: _____

Company: _____

Address: _____

City/State/Zip: _____

Telephone: _____

Fold Here

No Postage Necessary If Mailed In The United States

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 995 CHARLOTTESVILLE, VA

POSTAGE WILL BE PAID BY ADDRESSEE:

ATTENTION MANAGER TECHNICAL PUBLICATIONS
GE Fanuc Automation North America Inc
POBOX8106
CHARLOTTESVILLE VA 22907-6063

Fold Here

NOTES