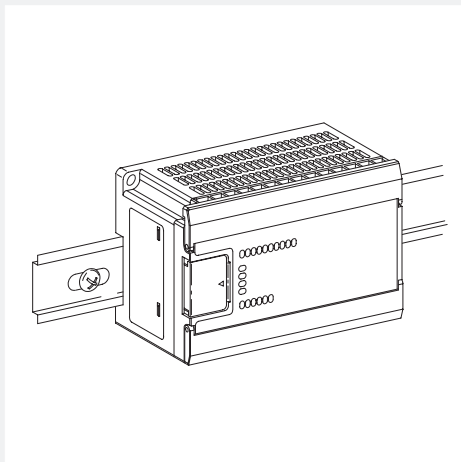




Allen-Bradley

***MicroLogix™ 1000
with Hand-Held
Programmer (HHP)***

(Cat. No. 1761-HHP-B30)



User Manual



Important User Information

Because of the variety of uses for the products described in this publication, those responsible for the application and use of this control equipment must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards.

The illustrations, charts, sample programs and layout examples shown in this guide are intended solely for purposes of example. Since there are many variables and requirements associated with any particular installation, Allen-Bradley does not assume responsibility or liability (to include intellectual property liability) for actual use based upon the examples shown in this publication.

Allen-Bradley publication SGI-1.1, *Safety Guidelines for the Application, Installation, and Maintenance of Solid-State Control* (available from your local Allen-Bradley office), describes some important differences between solid-state equipment and electromechanical devices that should be taken into consideration when applying products such as those described in this publication.

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

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



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

Attention statements help you to:

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

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

Preface

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

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

Who Should Use this Manual

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

You should have a basic understanding of electrical circuitry and familiarity with relay logic. If you do not, obtain the proper training before using this product.

Purpose of this Manual

This manual is a reference guide for the MicroLogix™ 1000 Programmable Controller with a MicroLogix 1000 Hand-Held Programmer (HHP). It describes the procedures you use to install, wire, and program your micro controller. This manual:

- gives you an overview of the micro controller system
- provides a quick start chapter for beginners
- describes how to use the Hand-Held Programmer
- guides you through how to interpret the instruction set
- contains application examples to show the instruction set in use

If you are using programming software with your MicroLogix 1000 Programmable Controller, see page P-4 for related publications.

Contents of this Manual

| Tab | Chapter | Title | Contents |
|-------------|---------|---|---|
| | | Preface | Describes the purpose, background, and scope of this manual. Also specifies the audience for whom this manual is intended. |
| Installing | 1 | Installing Your Controller | Provides controller installation procedures and system safety considerations. |
| | 2 | Wiring Your Controller | Provides wiring guidelines and diagrams. |
| | 3 | Connecting the System | Gives information on wiring your controller system for the DF1 protocol or DH-485 network. |
| Programming | 4 | Using Your Hand-Held Programmer | Describes how to power-up and use your MicroLogix 1000 Hand-Held Programmer (HHP). Also explains how to install the HHP's memory module. |
| | 5 | Quick Start for New Users | Provides step-by-step instructions on how to enter a program, edit it, and then monitor it. |
| | 6 | Programming Overview | Provides an overview of principles of machine control, a section on file organization and addressing, and a program development model. |
| | 7 | Using Analog | Provides information on I/O image file format, I/O configuration, input filter and update times and conversion of analog data. |
| | 8 | Using Basic Instructions | Describes how to use the instructions for relay replacement functions, counting, and timing. |
| | 9 | Using Comparison Instructions | Describes how to use the instructions to compare values of data in your logic program. |
| | 10 | Using Math Instructions | Describes how to use the instructions that perform basic math functions. |
| | 11 | Using Data Handling Instructions | Describes how to perform data handling instructions, including move and logical instructions and FIFO and LIFO instructions. |
| | 12 | Using Program Flow Control Instructions | Describes the instructions that affect program flow and execution. |
| | 13 | Using Application Specific Instructions | Describes the bit shift, sequencer and STI related instructions. |
| | 14 | Using High-Speed Counter Instructions | Describes the four modes of the high-speed counter instruction and its related instructions. |
| | 15 | Using Communication Protocols | Provides a general overview of the types of communication, and explains how to establish network communication using the message instruction. |

| Tab | Chapter | Title | Contents |
|-----------------|------------|--|---|
| Programming | 16 | Instruction List Programming | Provides examples to teach you Instruction List programming and describes programming considerations. |
| | 17 | Entering and Editing Your Program | Describes the various editing functions you can use with your program, including search, overwrite, and delete. |
| | 18 | After You've Entered Your Program | Describes how to configure, run, and monitor your program. |
| | 19 | Common Procedures | Describes how to perform additional procedures using the HHP menu. |
| Troubleshooting | 20 | Troubleshooting Your System | Explains how to interpret and correct problems with your micro controller system. |
| Reference | Appendix A | Hardware Reference | Provides physical, electrical, environmental, and functional specifications. |
| | Appendix B | Programming Reference | Explains the system status file, lists the HHP function codes, and provides instruction execution times. |
| | Appendix C | Valid Addressing Modes and File Types for Instruction Parameters | Provides a listing of the instructions along with their parameters and valid file types. |
| | Appendix D | Understanding the Communication Protocols | Contains descriptions of the DF1 protocol and DH-485 network. |
| Reference | Appendix E | Application Programs | Provides advanced application examples for the high-speed counter, sequencer, and bit shift instructions. |
| Reference | Appendix F | Optional Analog Input Software Calibration | Explains how to calibrate your controller using software offsets. |
| | | Glossary | Contains definitions for terms and abbreviations that are specific to this product. |



For More Information

As part of our effort to preserve, protect, and improve our environment, Allen-Bradley is reducing the amount of paper we use. Less paper means more options for you. In addition to traditional printed publications and CD-ROM versions, we now offer on-line manuals with the most up-to-date information you can get. We recommend that you read the related publications listed on the next page before starting up your control system.

Related Publications

| For | Read this Document | Document Number |
|--|---|-----------------|
| A description on how to install and use your MicroLogix™ 1000 Programmable Controllers. This manual also contains status file data and instruction set information | MicroLogix™ 1000 Programmable Controllers User Manual | 1761-6.3 |
| A reference manual that contains the status file data and the instruction set information for the SLC 500 processors and MicroLogix 1000 controllers | SLC 500™ and MicroLogix™ 1000 Instruction Set Reference Manual | 1747-6.15 |
| Information on mounting and wiring the MicroLogix 1000 controllers, including a mounting template for easy installation | MicroLogix™ 1000 Programmable Controllers Installation Instructions | 1761-5.1.2 |
| | MicroLogix™ 1000 (Analog) Programmable Controllers Installation Instructions | 1761-5.1.3 |
| The procedures necessary to install and connect the AIC+ and DNI | Advanced Interface Converter (AIC+) and DeviceNet Interface (DNI) Installation Instructions | 1761-5.11 |
| A description on how to install and connect an AIC+. This manual also contains information on network wiring. | Advanced Interface Converter (AIC+) User Manual | 1761-6.4 |
| Information on how to install, configure, and commission a DNI | DeviceNet Interface™ User Manual | 1761-6.5 |
| In-depth information on grounding and wiring Allen-Bradley programmable controllers | Allen-Bradley Programmable Controller Grounding and Wiring Guidelines | 1770-4.1 |

How to Get More Information

| For | Obtain Information By |
|--|--|
| Fast access to related publications | <ul style="list-style-type: none"> • Visiting the MicroLogix internet site http://www.abmicrologix.com — Electronic versions of our manuals are available for you to search and download. • Calling local Allen-Bradley distributor. |
| Publications in printed or CD-ROM format | Ordering a manual or CD-ROM using one of the following methods: <ul style="list-style-type: none"> • Fill out and return the User Manual Request Card that was shipped with the unit. • Visiting the Automation Bookstore at http://www.theautomationbookstore.com |
| Multiple copies of a manual | <ul style="list-style-type: none"> • Visiting the Automation Bookstore at http://www.theautomationbookstore.com |
| Manuals in other languages | Adding a 2-letter suffix to the end of the publication number when ordering. <ul style="list-style-type: none"> • French – FR • German – DE • Italian – IT • Spanish – ES • Portuguese – PT (DNI only) |


Related Documentation

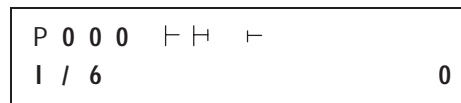
The following documents contain additional information concerning Allen-Bradley products. To obtain a copy, contact your local Allen-Bradley office or distributor.

| For | Read This Document | Document Number |
|---|---|--|
| A description of important differences between solid-state programmable controller products and hard-wired electromechanical devices | Application Considerations for Solid-State Controls | SGI-1.1 |
| An article on wire sizes and types for grounding electrical equipment | National Electrical Code | Published by the National Fire Protection Association of Boston, MA. |
| A complete listing of current documentation, including ordering instructions. Also indicates whether the documents are available on CD-ROM or in multi-languages. | Allen-Bradley Publication Index | SD499 |
| A glossary of industrial automation terms and abbreviations | Allen-Bradley Industrial Automation Glossary | AG-7.1 |

Common Techniques Used in this Manual


The following conventions are used throughout this manual:

- Bulleted lists such as this one provide information, not procedural steps.
- Numbered lists provide sequential steps or hierarchical information.
- *Italic* type is used for emphasis.
- Text in this font indicates words that appear on the HHP display.
-  Keypad icons, like the one at the left, match the key you should press.
- For operations that require you to press a sequence of keys, the keypad icons are displayed horizontally on the page, with the resulting screen shown beneath. For example:



- If a character is flashing on the HHP display, it is shown unbolded (such as the P in the screen above).
- For operations that require you to press two keys simultaneously, the keypad icons are displayed side-by-side as shown here:



-  For operations that require you to press an arrow key, the key you should press is shown bolded, such as the right arrow key shown here.

Allen-Bradley Support

Allen-Bradley offers support services worldwide, with over 75 Sales/Support Offices, 512 authorized Distributors and 260 authorized Systems Integrators located throughout the United States alone, plus Allen-Bradley representatives in every major country in the world.

Local Product Support

Contact your local Allen-Bradley representative for:

- sales and order support
- product technical training
- warranty support
- support service agreements

Technical Product Assistance

If you need to contact Allen-Bradley for technical assistance, please review the information in the *Troubleshooting* chapter first. Then call your local Allen-Bradley representative.

Your Questions or Comments on this Manual

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

If you have any suggestions for how this manual could be made more useful to you, please contact us at the address below:

Allen-Bradley Company, Inc.
Control and Information Group
Technical Communication, Dept. 602V, T122
P.O. Box 2086
Milwaukee, WI 53201-2086

or visit our internet page at:

<http://www.abmicrologix.com>

Preface

| | |
|---|-----|
| Who Should Use this Manual | P-1 |
| Purpose of this Manual | P-1 |
| Common Techniques Used in this Manual | P-5 |
| Allen-Bradley Support | P-6 |

Hardware

Installing Your Controller

Chapter 1

| | |
|---|------|
| Compliance to European Union Directives | 1-1 |
| Hardware Overview | 1-2 |
| Master Control Relay | 1-3 |
| Using Surge Suppressors | 1-7 |
| Safety Considerations | 1-9 |
| Power Considerations | 1-10 |
| Preventing Excessive Heat | 1-11 |
| Controller Spacing | 1-12 |
| Mounting the Controller | 1-12 |

Wiring Your Controller

Chapter 2

| | |
|--|------|
| Grounding Guidelines | 2-1 |
| Sinking and Sourcing Circuits | 2-2 |
| Wiring Recommendations | 2-3 |
| Wiring Diagrams, Discrete Input and Output Voltage Ranges | 2-6 |
| Minimizing Electrical Noise on Analog Controllers | 2-20 |
| Grounding Your Analog Cable | 2-20 |
| Wiring Your Analog Channels | 2-21 |
| Analog Voltage and Current Input and Output Ranges | 2-22 |
| Wiring Your Controller for High-Speed Counter Applications | 2-23 |

Connecting the System

Chapter 3

| | |
|--|------|
| Connecting the HHP | 3-1 |
| Connecting to a DH-485 Network | 3-3 |
| Connecting the AIC+ | 3-6 |
| Establishing Communication | 3-12 |
| DeviceNet Communications | 3-13 |

Programming

Using Your Hand-Held Programmer

Chapter 4

| | |
|---|------|
| About Your HHP | 4-1 |
| Installing the Optional Memory Module | 4-3 |
| The Keys You Use | 4-4 |
| Identifying the Power-Up Sequence | 4-6 |
| Understanding the HHPs Functional Areas | 4-7 |
| Changing the HHPs Defaults | 4-17 |

Quick Start for New Users

Chapter 5

| | |
|--|------|
| What to Do First | 5-1 |
| Preparing to Enter a New Program | 5-2 |
| Entering and Running the Program | 5-4 |
| Monitoring Operation | 5-9 |
| What to Do Next | 5-12 |

Programming Overview

Chapter 6

| | |
|--|------|
| Principles of Machine Control | 6-1 |
| Understanding File Organization | 6-3 |
| Understanding How Programs are Stored and Accessed | 6-5 |
| Addressing Data Files | 6-7 |
| Applying Logic to Your Schematics | 6-11 |
| Developing Your Logic Program – A Model | 6-17 |

Using Analog

Chapter 7

| | |
|-------------------------------------|-----|
| I/O Image | 7-1 |
| I/O Configuration | 7-2 |
| Input Filter and Update Times | 7-2 |
| Converting Analog Data | 7-4 |

Using Basic Instructions

Chapter 8

| | |
|--|-----|
| About Basic Instructions | 8-2 |
| Bit Instructions Overview | 8-3 |
| Load (LD), And (AND), and Or (OR) | 8-3 |
| Load Inverted (LDI), And Inverted (ANI), and Or Inverted (ORI) | 8-4 |
| Load True (LDT) and Or True (ORT) | 8-6 |
| One-Shot Rising (OSR) | 8-7 |

| | |
|--|------|
| Output (OUT) | 8-8 |
| Set (SET) and Reset (RST) | 8-8 |
| Branch Instructions Overview | 8-9 |
| Memory Push (MPS), Memory Read (MRD), and Memory Pop (MPP) | 8-10 |
| And Block (ANB) and Or Block (ORB) | 8-12 |
| Timer Instructions Overview | 8-14 |
| Timer On-Delay (TON) | 8-16 |
| Timer Off-Delay (TOF) | 8-18 |
| Retentive Timer (RTO) | 8-20 |
| Counter Instructions Overview | 8-21 |
| Count Up (CTU) | 8-24 |
| Count Down (CTD) | 8-25 |
| Reset (RES) | 8-27 |
| Basic Instructions in the Paper Drilling Machine Application Example | 8-28 |

Using Comparison Instructions

Chapter 9

| | |
|---|------|
| About the Comparison Instructions | 9-2 |
| Comparison Instructions Overview | 9-2 |
| Equal (EQU) | 9-3 |
| Not Equal (NEQ) | 9-4 |
| Less Than (LES) | 9-5 |
| Less Than or Equal (LEQ) | 9-6 |
| Greater Than (GRT) | 9-7 |
| Greater Than or Equal (GEO) | 9-8 |
| Masked Comparison for Equal (MEQ) | 9-9 |
| Limit Test (LIM) | 9-10 |
| Comparison Instructions in the Paper Drilling Machine Application Example | 9-12 |

Using Math Instructions

Chapter 10

| | |
|---|-------|
| About the Math Instructions | 10-1 |
| Math Instructions Overview | 10-2 |
| Add (ADD) | 10-4 |
| Subtract (SUB) | 10-5 |
| 32-Bit Addition and Subtraction | 10-6 |
| Multiply (MUL) | 10-8 |
| Divide (DIV) | 10-9 |
| Double Divide (DDV) | 10-10 |
| Clear (CLR) | 10-11 |
| Square Root (SQR) | 10-11 |
| Scale Data (SCL) | 10-12 |
| Math Instructions in the Paper Drilling Machine Application Example | 10-15 |

Using Data Handling Instructions

Chapter 11

| | |
|---|-------|
| About the Data Handling Instructions | 11-2 |
| Convert to BCD (TOD) | 11-2 |
| Convert from BCD (FRD) | 11-3 |
| Decode 4 to 1 of 16 (DCD) | 11-7 |
| Encode 1 of 16 to 4 (ENC) | 11-8 |
| Copy File (COP) and Fill File (FLL) Instructions | 11-10 |
| Move and Logical Instructions Overview | 11-13 |
| Move (MOV) | 11-15 |
| Masked Move (MVM) | 11-16 |
| And (AND) | 11-18 |
| Or (OR) | 11-19 |
| Exclusive Or (XOR) | 11-20 |
| Not (NOT) | 11-21 |
| Negate (NEG) | 11-22 |
| FIFO and LIFO Instructions Overview | 11-23 |
| FIFO Load (FFL) and FIFO Unload (FFU) | 11-25 |
| LIFO Load (LFL) and LIFO Unload (LFU) | 11-28 |
| Data Handling Instructions in the Paper Drilling Machine Application Example | 11-31 |

Using Program Flow Control Instructions

Chapter 12

| | |
|--|-------|
| About the Program Flow Control Instructions | 12-1 |
| Jump (JMP) and Label (LBL) | 12-2 |
| Jump to Subroutine (JSR), Subroutine (SBR), and Return (RET) | 12-3 |
| Master Control Reset (MCR) | 12-6 |
| Temporary End (TND) | 12-7 |
| Suspend (SUS) | 12-7 |
| Immediate Input with Mask (IIM) | 12-8 |
| Immediate Output with Mask (IOM) | 12-9 |
| Program Flow Control Instructions in the Paper Drilling Machine Application Example | 12-10 |

Using Application Specific Instructions

Chapter 13

| | |
|--|-------|
| About the Application Specific Instructions | 13-1 |
| Bit Shift Instructions Overview | 13-2 |
| Bit Shift Left (BSL) | 13-3 |
| Bit Shift Right (BSR) | 13-4 |
| Sequencer Instructions Overview | 13-6 |
| Sequencer Output (SQO) and Sequencer Compare (SQC) | 13-6 |
| Sequencer Load (SQL) | 13-12 |
| Selectable Timed Interrupt (STI) Function Overview | 13-15 |

| | |
|--|-------|
| Selectable Timed Disable (STD) and Enable (STE) | 13-17 |
| Selectable Timed Start (STS) | 13-18 |
| Interrupt Subroutine (INT) | 13-19 |
| Application Specific Instructions in the Paper Drilling Machine Application Example | 13-20 |

Using High-Speed Counter Instructions

Chapter 14

| | |
|--|-------|
| About the High-Speed Counter Instructions | 14-1 |
| High-Speed Counter Instructions Overview | 14-2 |
| High-Speed Counter (HSC) | 14-4 |
| High-Speed Counter Load (HSL) | 14-15 |
| High-Speed Counter Reset (RES) | 14-19 |
| High-Speed Counter Reset Accumulator (RAC) | 14-20 |
| High-Speed Counter Interrupt Enable (HSE) and Disable (HSD) | 14-21 |
| Update High-Speed Counter Image Accumulator (OUT) | 14-23 |
| What Happens to the HSC When Going to RRUN Mode | 14-23 |
| High-Speed Counter Instructions in the Paper Drilling Machine Application Example | 14-28 |

Using Communication Protocols

Chapter 15

| | |
|---|-------|
| Types of Communication | 15-1 |
| Message Instruction (MSG) | 15-2 |
| Timing Diagram for a Successful MSG Instruction | 15-9 |
| MSG Instruction Error Codes | 15-11 |
| Application Examples that Use the MSG Instruction | 15-12 |

Instruction List Programming Concepts

Chapter 16

| | |
|--------------------------------------|------|
| Programming Examples | 16-1 |
| Programming Considerations | 16-8 |

Entering and Editing Your Program

Chapter 17

| | |
|--|------|
| Entering the Program Monitor | 17-1 |
| Editing Considerations | 17-3 |
| Editing Modes | 17-3 |
| Deleting Instructions and Rungs | 17-6 |
| Searching for Specific Addresses | 17-8 |

After You've Entered Your Program

Chapter 18

Changing the Program Configuration Defaults 18-1
Accepting Your Program Edits 18-20
Changing Controller Modes 18-20
Monitoring Your Controller 18-24
Viewing Data Table Files 18-27
Using the Multi-Point Function 18-30
Forcing Inputs and Outputs 18-34

Common Procedures

Chapter 19

Using a Memory Module 19-1
Clearing a Program from the Micro Controller 19-6
Changing the Micro Controller's Baud Rate 19-6
Changing the Micro Controller's Communication Defaults 19-7

Troubleshooting

Troubleshooting Your System

Chapter 20

Understanding the Controller LED Status 20-1
Identifying HHP Errors 20-3
Using the Trace Feature 20-8
Controller Error Recovery Model 20-10
Identifying Controller Faults 20-11
Recovering Your Work 20-16
Calling Allen-Bradley for Assistance 20-16

Reference

Hardware Reference

Appendix A

Controller Specifications A-1
Controller Dimensions A-7
Hand-Held Programmer Specifications A-9
Controller and Hand-Held Programmer Accessories and Replacement Parts A-11

Programming Reference**Appendix B**

| | |
|--|------|
| Controller Status File | B-1 |
| Function Codes | B-13 |
| Instruction Execution Times and Memory Usage | B-16 |

Valid Addressing Modes and File Types for Instruction Parameters**Appendix C**

| | |
|----------------------------------|-----|
| Available File Types | C-1 |
| Available Addressing Modes | C-2 |

Understanding the Communication Protocols**Appendix D**

| | |
|--------------------------------------|-----|
| RS-232 Communication Interface | D-1 |
| DF1 Full-Duplex Protocol | D-2 |
| DF1 Half-Duplex Slave Protocol | D-4 |
| DH-485 Communication Protocol | D-9 |

Application Example Programs**Appendix E**

| | |
|--|------|
| Paper Drilling Machine Application Example | E-2 |
| Time Driven Sequencer Application Example | E-25 |
| Event Driven Sequencer Application Example | E-27 |
| Bottle Line Example | E-29 |

Optional Analog Input Software Calibration**Appendix F**

| | |
|---|-----|
| Calibrating an Analog Input Channel | F-1 |
|---|-----|

Glossary

Summary of Changes

The information below summarizes the changes to this manual since the last printing as Publication 1761-6.2—October 1997.

To help you find new information and updated information in this release of the manual, we have included change bars as shown to the right of this paragraph.

New Information

The table below lists sections that document new features and additional information about existing features, and shows where to find this new information.

| For This New Information | See |
|-------------------------------------|------------------------------------|
| Power supply inrush | page 1-11 |
| Class I, Division 2 certification | pages 1-12, A-2 |
| analog controllers | pages 2-17, 7-1, 18-14, appendix A |
| automatic protocol switching | page 3-13 |
| DeviceNet communications | page 3-13 |
| software compatibility | page 4-1 |
| SCL instruction application example | page 10-14 |
| remote network support | page D-17 |

Updated Information

Changes from the previous release of this manual that require you to reference information differently are as follows:

- The safety considerations for mounting your controller have been updated; see chapter 1, Installing Your Controller.
- The section on establishing communication has been updated; see chapter 3, Connecting the System.
- For updated information on HHP support and compatibility of the series functionality of your MicroLogix controller, see chapter 15, Using Communication Protocols.
- The message timing diagram has been updated; see chapter 15, Using Communication Protocols.
- The MicroLogix 1000 programmable controllers' VA ratings and power supply inrush specifications have been updated; see appendix A, Hardware Reference.
- The agency certification specifications have been updated; see appendix A, Hardware Reference.
- The analog output overall accuracy specification has been updated; see appendix A, Hardware Reference.
- The user interrupt latency information has been updated; see appendix B, Programming Reference.
- The DF1 Full-Duplex and DH-485 configuration parameters have been updated; see appendix D, Understanding Communication Protocols.

Installing Your Controller

This chapter shows you how to install your MicroLogix 1000 Programmable Controller. The only tools you require are a Flat head or Phillips head screwdriver and drill. Topics include:

- compliance to European Union Directives
- hardware overview
- master control relay
- surge suppressors
- safety considerations
- power considerations
- preventing excessive heat
- controller spacing
- mounting the controller

Compliance to European Union Directives

If this product has the CE mark it is approved for installation within the European Union and EEA regions. It has been designed and tested to meet the following directives.

EMC Directive

This product is tested to meet Council Directive 89/336/EEC Electromagnetic Compatibility (EMC) and the following standards, in whole or in part, documented in a technical construction file:

- EN 50081-2
EMC – Generic Emission Standard, Part 2 – Industrial Environment
- EN 50082-2
EMC – Generic Immunity Standard, Part 2 – Industrial Environment

This product is intended for use in an industrial environment.

Low Voltage Directive

This product is tested to meet Council Directive 73/23/EEC Low Voltage, by applying the safety requirements of EN 61131-2 Programmable Controllers, Part 2 – Equipment Requirements and Tests.

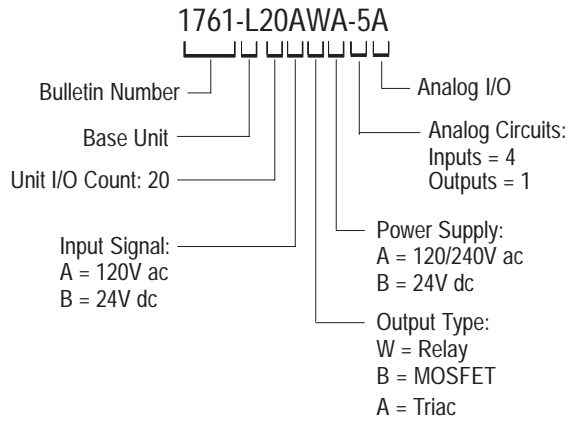
For specific information required by EN 61131-2, see the appropriate sections in this publication, as well as the following Allen-Bradley publications:

- Industrial Automation Wiring and Grounding Guidelines For Noise Immunity, publication 1770-4.1
- Guidelines for Handling Lithium Batteries, publication AG-5.4
- Automation Systems Catalog, publication B111

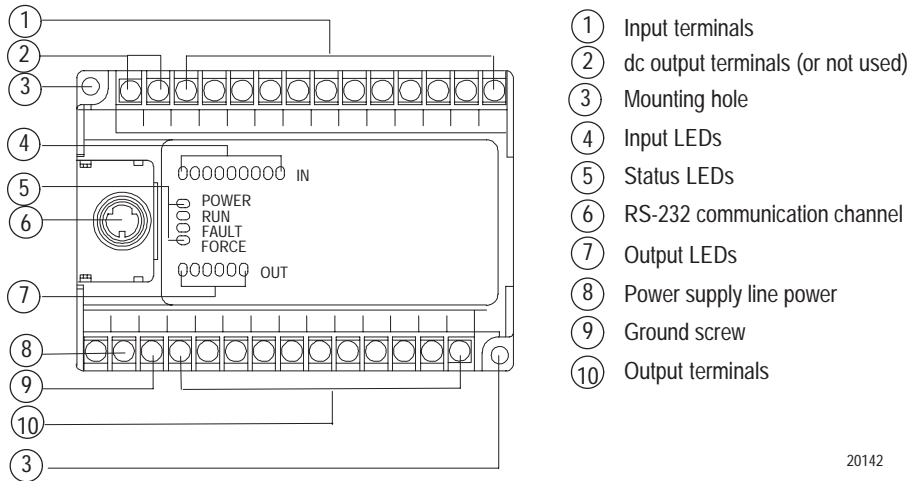
Hardware Overview

The MicroLogix 1000 programmable controller is a packaged controller containing a power supply, input circuits, output circuits, and a processor. The controller is available in 10 I/O, 16 I/O and 32 I/O configurations, as well as an analog version with 20 discrete I/O and 5 analog I/O.

The catalog number for the controller is composed of the following:



The hardware features of the controller are:



20142

Master Control Relay

A hard-wired master control relay (MCR) provides a reliable means for emergency controller shutdown. Since the master control relay allows the placement of several emergency-stop switches in different locations, its installation is important from a safety standpoint. Overtravel limit switches or mushroom head push buttons are wired in series so that when any of them opens, the master control relay is de-energized. This removes power to input and output device circuits. Refer to the figure on page 1–5.



ATTENTION: Never alter these circuits to defeat their function, since serious injury and/or machine damage could result.

Important: If you are using an external dc output power supply, interrupt the dc output side rather than the ac line side of the supply to avoid the additional delay of power supply turn-off.

The external ac line of the dc output power supply should be fused.

Connect a set of master control relays in series with the dc power supplying the input and output circuits.

Place the main power disconnect switch where operators and maintenance personnel have quick and easy access to it. If you mount a disconnect switch inside the controller enclosure, place the switch operating handle on the outside of the enclosure, so that you can disconnect power without opening the enclosure.

Whenever any of the emergency-stop switches are opened, power to input and output devices should be removed.

When you use the master control relay to remove power from the external I/O circuits, power continues to be provided to the controller's power supply so that diagnostic indicators on the processor can still be observed.

The master control relay is not a substitute for a disconnect to the controller. It is intended for any situation where the operator must quickly de-energize I/O devices only. When inspecting or installing terminal connections, replacing output fuses, or working on equipment within the enclosure, use the disconnect to shut off power to the rest of the system.

Important: Do not control the master control relay with the controller. Provide the operator with the safety of a direct connection between an emergency-stop switch and the master control relay.

Using Emergency-Stop Switches

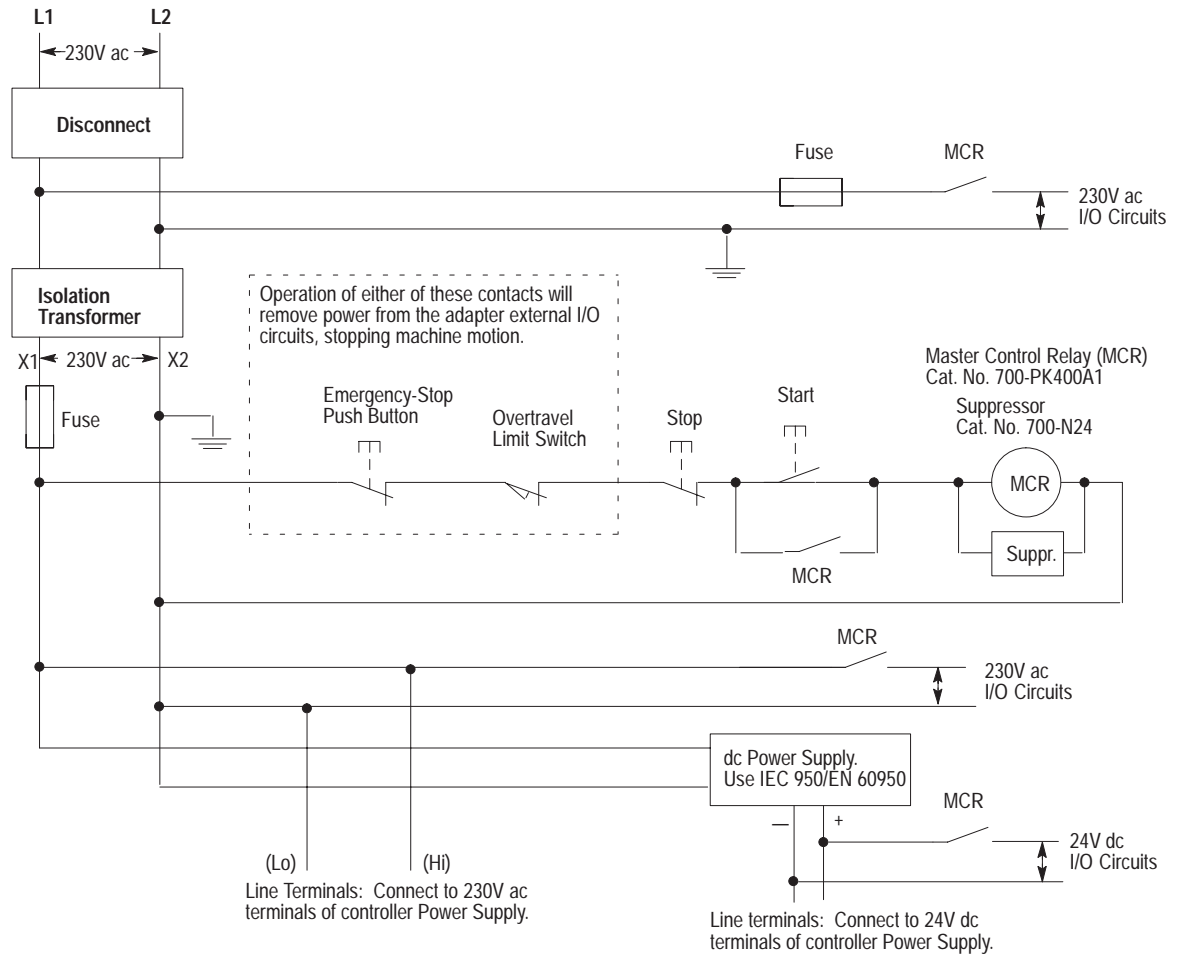
When using emergency-stop switches, adhere to the following points:

- Do not program emergency-stop switches in the controller program. Any emergency-stop switch should turn off all machine power by turning off the master control relay.
- Observe all applicable local codes concerning the placement and labeling of emergency-stop switches.
- Install emergency-stop switches and the master control relay in your system. Make certain that relay contacts have a sufficient rating for your application. Emergency-stop switches must be easy to reach.
- In the following illustration, input and output circuits are shown with MCR protection. However, in most applications, only output circuits require MCR protection.

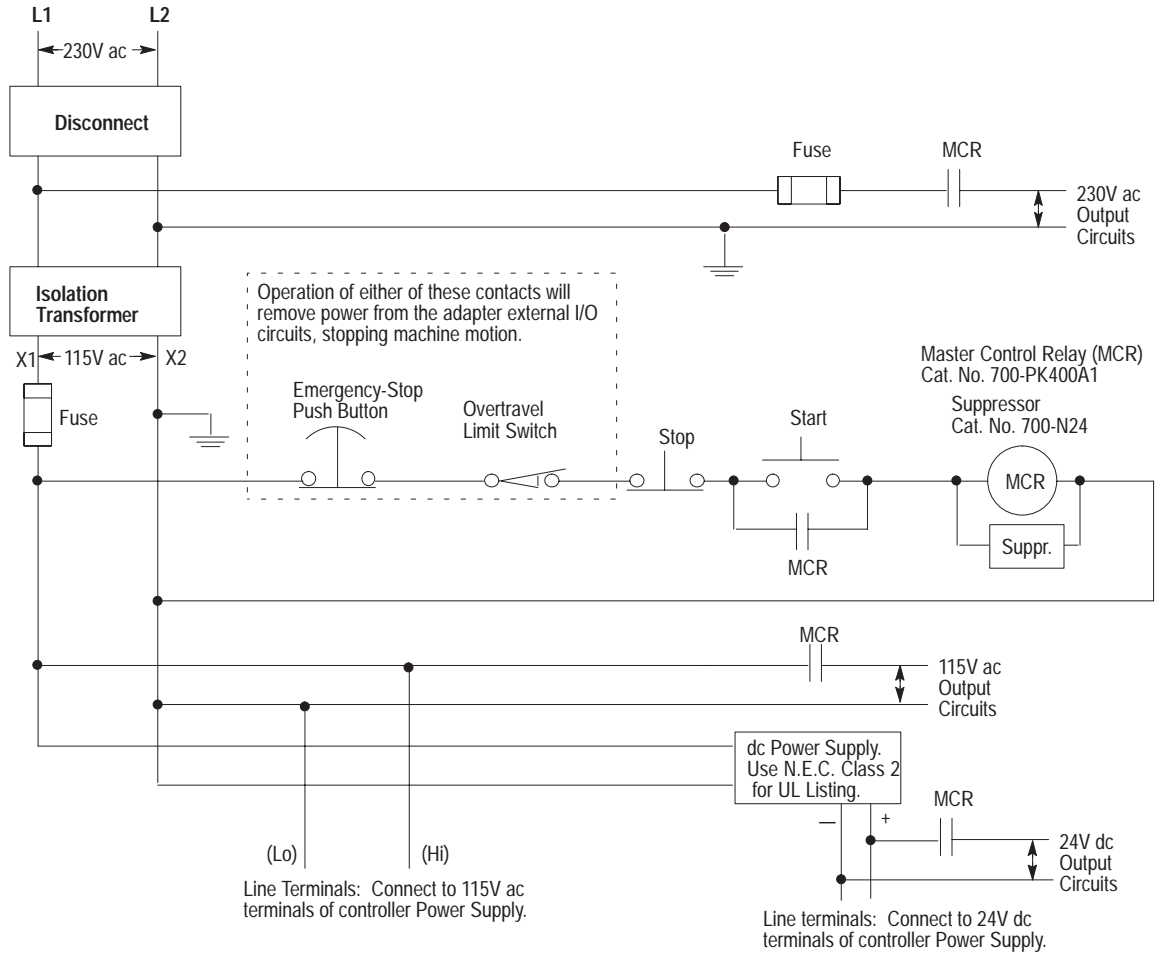
The following illustrations show the Master Control Relay wired in a grounded system.

Important: The illustrations only show output circuits with MCR protection. In most applications input circuits do not require MCR protection; however, if you need to remove power from all field devices, you must include MCR contacts in series with input power wiring.

Schematic (Using IEC Symbols)



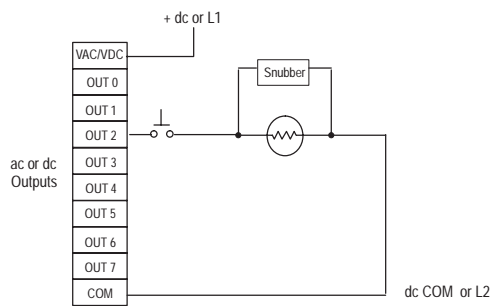
Schematic (Using ANSI/CSA Symbols)



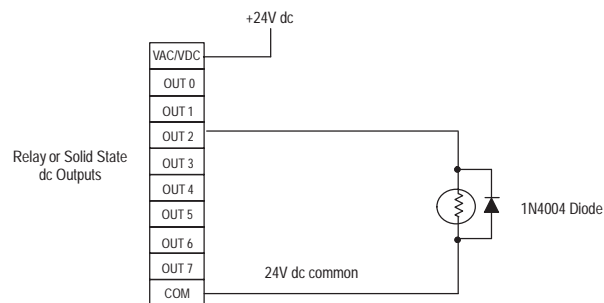
Using Surge Suppressors

Inductive load devices such as motor starters and solenoids require the use of some type of surge suppression to protect the controller output contacts. Switching inductive loads without surge suppression can *significantly* reduce the lifetime of relay contacts. By adding a suppression device directly across the coil of an inductive device, you will prolong the life of the switch contacts. You also reduce the effects of voltage transients caused by interrupting the current to that inductive device, and prevent electrical noise from radiating into system wiring.

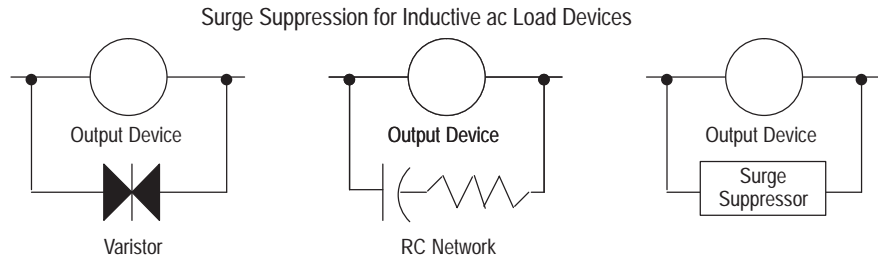
The following diagram shows an output with a suppression device. We recommend that you locate the suppression device as close as possible to the load device.



If you connect a micro controller FET output to an inductive load, we recommend that you use a 1N4004 diode for surge suppression, as shown in the following illustration.



Suitable surge suppression methods for inductive ac load devices include a varistor, an RC network, or an Allen-Bradley surge suppressor, all shown below. These components must be appropriately rated to suppress the switching transient characteristic of the particular inductive device. See the table on page 1-9 for recommended suppressors.

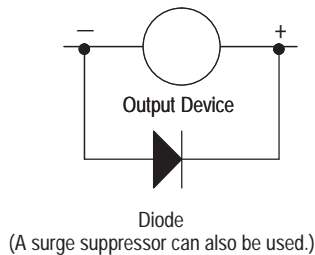


If you connect a micro controller triac output to control an inductive load, we recommend that you use varistors to suppress noise. Choose a varistor that is appropriate for the application. The suppressors we recommend for triac outputs when switching 120V ac inductive loads are a Harris MOV, part number V175 LA10A, or an Allen-Bradley MOV, catalog number 599-K04 or 599-KA04. Consult the varistor manufacturer's data sheet when selecting a varistor for your application.

For inductive dc load devices, a diode is suitable. A 1N4004 diode is acceptable for most applications. A surge suppressor can also be used. See the table on page 1-9 for recommended suppressors.

As shown in the illustration below, these surge suppression circuits connect directly across the load device. This reduces arcing of the output contacts. (High transient can cause arcing that occurs when switching off an inductive device.)

Surge Suppression for Inductive dc Load Devices



Recommended Surge Suppressors

We recommend the Allen-Bradley surge suppressors shown in the following table for use with Allen-Bradley relays, contactors, and starters.

| Device | Coil Voltage | Suppressor Catalog Number |
|---|----------------------------|---------------------------|
| Bulletin 509 Motor Starter Bulletin 509 Motor Starter | 120V ac 240V ac | 599-K04 599-KA04 |
| Bulletin 100 Contactor Bulletin 100 Contactor | 120V ac 240V ac | 199-FSMA1 199-FSMA2 |
| Bulletin 709 Motor Starter | 120V ac | 1401-N10 |
| Bulletin 700 Type R, RM Relays | ac coil | None Required |
| Bulletin 700 Type R Relay Bulletin 700 Type RM Relay | 12V dc 12V dc | 700-N22 700-N28 |
| Bulletin 700 Type R Relay Bulletin 700 Type RM Relay | 24V dc 24V dc | 700-N10 700-N13 |
| Bulletin 700 Type R Relay Bulletin 700 Type RM Relay | 48V dc 48V dc | 700-N16 700-N17 |
| Bulletin 700 Type R Relay Bulletin 700 Type RM Relay | 115-125V dc 115-125V dc | 700-N11 700-N14 |
| Bulletin 700 Type R Relay Bulletin 700 Type RM Relay | 230-250V dc 230-250V dc | 700-N12 700-N15 |
| Bulletin 700 Type N, P, or PK Relay | 150V max, ac or DC | 700-N24 |
| Miscellaneous electromagnetic devices limited to 35 sealed VA | 150V max, ac or DC | 700-N24 |

Safety Considerations

Safety considerations are an important element of proper system installation. Actively thinking about the safety of yourself and others, as well as the condition of your equipment, is of primary importance. We recommend reviewing the following safety considerations.

Disconnecting Main Power



ATTENTION: Explosion Hazard — Do not replace components or disconnect equipment unless power has been switched off and the area is known to be non-hazardous.

The main power disconnect switch should be located where operators and maintenance personnel have quick and easy access to it. In addition to disconnecting electrical power, all other sources of power (pneumatic and hydraulic) should be de-energized before working on a machine or process controlled by a controller.



ATTENTION: Explosion Hazard — Do not connect or disconnect while circuit is live unless area is known to be non-hazardous.

Safety Circuits

Circuits installed on the machine for safety reasons, like overtravel limit switches, stop push buttons, and interlocks, should always be hard-wired directly to the master control relay. These devices must be wired in series so that when any one device opens, the master control relay is de-energized thereby removing power to the machine. Never alter these circuits to defeat their function. Serious injury or machine damage could result.

Power Distribution

There are some points about power distribution that you should know:

- The master control relay must be able to inhibit all machine motion by removing power to the machine I/O devices when the relay is de-energized.
- If you are using a dc power supply, interrupt the load side rather than the ac line power. This avoids the additional delay of power supply turn-off. The dc power supply should be powered directly from the fused secondary of the transformer. Power to the dc input and output circuits is connected through a set of master control relay contacts.

Periodic Tests of Master Control Relay Circuit

Any part can fail, including the switches in a master control relay circuit. The failure of one of these switches would most likely cause an open circuit, which would be a safe power-off failure. However, if one of these switches shorts out, it no longer provides any safety protection. These switches should be tested periodically to assure they will stop machine motion when needed.

Power Considerations

The following explains power considerations for the micro controllers.

Isolation Transformers

You may want to use an isolation transformer in the ac line to the controller. This type of transformer provides isolation from your power distribution system and is often used as a step down transformer to reduce line voltage. Any transformer used with the controller must have a sufficient power rating for its load. The power rating is expressed in volt-amperes (VA).

Power Supply Inrush

The MicroLogix power supply does not require or need a high inrush current. However, if the power source can supply a high inrush current, the MicroLogix power supply will accept it. There is a high level of inrush current when a large capacitor on the input of the MicroLogix is charged up quickly.

If the power source cannot supply high inrush current, the only effect is that the MicroLogix input capacitor charges up more slowly. The following considerations determine whether the power source needs to supply a high inrush current:

- power-up sequence of devices in system
- power source sag if it cannot source inrush current
- the effect of the voltage sag on other equipment

If the power source cannot provide high inrush current when the entire system in an application is powered, the MicroLogix powers-up more slowly. If part of an application's system is already powered and operating when the MicroLogix is powered, the source voltage may sag while the MicroLogix input capacitor is charging. A power source voltage sag can affect other equipment connected to the same power source. For example, a voltage sag may reset a computer connected to the same power source.

Loss of Power Source

The power supply is designed to withstand brief power losses without affecting the operation of the system. The time the system is operational during power loss is called "program scan hold-up time after loss of power." The duration of the power supply hold-up time depends on the type and state of the I/O, but is typically between 20 milliseconds and 3 seconds. When the duration of power loss reaches this limit, the power supply signals the processor that it can no longer provide adequate dc power to the system. This is referred to as a power supply shutdown.

Input States on Power Down

The power supply hold-up time as described above is generally longer than the turn-on and turn-off times of the inputs. Because of this, the input state change from "On" to "Off" that occurs when power is removed may be recorded by the processor before the power supply shuts down the system. The user program should be written to take this effect into account.

Other Types of Line Conditions

Occasionally the power source to the system can be temporarily interrupted. It is also possible that the voltage level may drop substantially below the normal line voltage range for a period of time. Both of these conditions are considered to be a loss of power for the system.

Preventing Excessive Heat

For most applications, normal convective cooling keeps the controller within the specified operating range. Ensure that the specified operating range is maintained. Proper spacing of components within an enclosure is usually sufficient for heat dissipation.

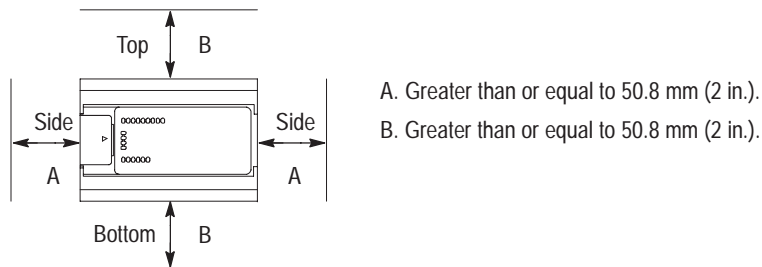
In some applications, a substantial amount of heat is produced by other equipment inside or outside the enclosure. In this case, place blower fans inside the enclosure to assist in air circulation and to reduce “hot spots” near the controller.

Additional cooling provisions might be necessary when high ambient temperatures are encountered.

Important: Do not bring in unfiltered outside air. Place the controller in an enclosure to protect it from a corrosive atmosphere. Harmful contaminants or dirt could cause improper operation or damage to components. In extreme cases, you may need to use air conditioning to protect against heat build-up within the enclosure.

Controller Spacing

The following figure shows the recommended *minimum* spacing for the controller. (Refer to appendix A for controller dimensions.)



20142

Mounting the Controller

This equipment is suitable for Class I, Division 2, Groups A, B, C, D or non-hazardous locations only, when product or packaging is marked.



ATTENTION – Explosion Hazard:

- Substitution of components may impair suitability for Class I, Division 2.
- This product must be installed in an enclosure. All cables connected to the product must remain in the enclosure or be protected by conduit or other means.

The controller should be mounted horizontally within an enclosure, using a DIN rail or mounting screws. Copy the template from page A-8 to help you space and mount the controller properly.



ATTENTION: Be careful of metal chips when drilling mounting holes for your controller. Drilled fragments that fall into the controller could cause damage. Do not drill holes above a mounted controller if the protective wrap is removed.

Use only the following communication cables in Class I, Division 2 Hazardous Locations.

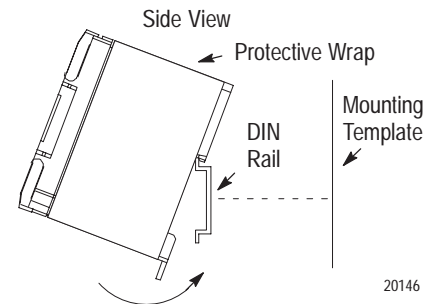
| Environment Classification | Communication Cable |
|---|------------------------|
| Class I, Division 2 Hazardous Environment | 1761-CBL-PM02 Series C |
| | 1761-CBL-HM02 Series C |
| | 1761-CBL-AM00 Series C |
| | 1761-CBL-AP00 Series C |
| | 2707-NC8 Series B |
| | 2707-NC9 Series B |
| | 2707-NC10 Series B |
| | 2707-NC11 Series B |

Using a DIN Rail

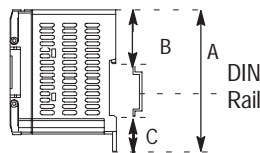
Use 35 mm (1.38 in.) DIN rails, such as item number 199-DR1 or 1492-DR5 from Bulletin 1492.

To install your controller on the DIN rail:

1. Mount your DIN rail. (Make sure that the placement of the controller on the DIN rail meets the recommended spacing requirements. Refer to controller dimensions in appendix A.)
2. Hook the top slot over the DIN rail.
3. While pressing the controller against the rail, snap the controller into position.
4. Leave the protective wrap attached until you are finished wiring the controller.



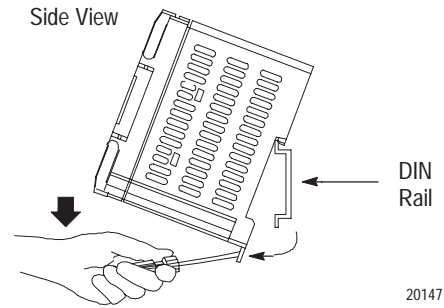
20146



| Call-out | Dimension |
|----------|-----------------|
| A | 84 mm (3.3 in.) |
| B | 33 mm (1.3 in.) |
| C | 16 mm (.63 in.) |

To remove your controller from the DIN rail:

1. Place a screwdriver in the DIN rail latch at the bottom of the controller.
2. Holding the controller, pry downward on the latch until the controller is released from the DIN rail.



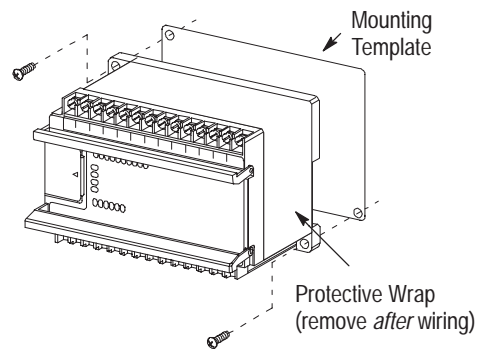
20147

Using Mounting Screws

To install your controller using mounting screws:

Important: Leave the protective wrap attached until you are finished wiring the controller.

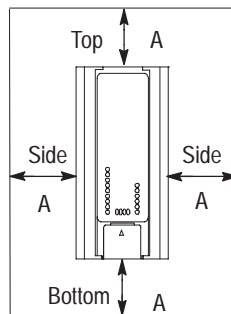
1. Use the mounting template from page A-8.
2. Secure the template to the mounting surface. (Make sure your controller is spaced properly.)
3. Drill holes through the template.
4. Remove the mounting template.
5. Mount the controller.



Mounting Your Controller Vertically

Your controller can also be mounted vertically within an enclosure using mounting screws or a DIN rail. To insure the stability of your controller, we recommend using mounting screws.

To insure the controller's reliability, the following environmental specifications must not be exceeded.



A. Greater than or equal to 50.8 mm (2 in.).

| Description: | Specification: |
|------------------------------------|--|
| Operating Temperature | 0°C to +40°C (+32°F to +113°F) ^① |
| Operating Shock (Panel mounted) | 9.0g peak acceleration (11±1 ms duration) 3 times each direction, each axis |
| Operating Shock (DIN rail mounted) | 7.0g peak acceleration (11±1 ms duration) 3 times each direction, each axis |

^① DC input voltage derated linearly from +30°C (30V to 26.4V).

Note: When mounting your controller vertically, the nameplate should be facing downward.


Wiring Your Controller

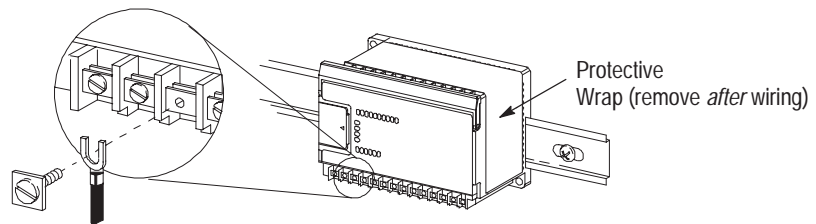
This chapter explains how to wire your MicroLogix 1000 Programmable Controller. Topics include:

- grounding guidelines
- sinking and sourcing circuits
- wiring recommendations
- wiring diagrams, input voltage ranges, and output voltage ranges

Grounding Guidelines

In solid-state control systems, grounding helps limit the effects of noise due to electromagnetic interference (EMI). Use the heaviest wire gauge listed for wiring your controller with a maximum length of 152.4 mm (6 in.). Run the ground connection from the ground screw of the controller (third screw from left on output terminal rung) to the ground bus.

Important:  This symbol denotes a functional earth ground terminal which provides a low impedance path between electrical circuits and earth for non-safety purposes, such as noise immunity improvement.



ATTENTION: All devices that connect to the user 24V power supply or to the RS-232 channel must be referenced to chassis ground or floating. Failure to follow this procedure may result in property damage or personal injury.



ATTENTION: Chassis ground, user 24V ground, and RS-232 ground are internally connected. You must connect the chassis ground terminal screw to chassis ground prior to connecting any devices.



ATTENTION: On the 1761-L10BWB, 1761-L16BWB, 1761-L16BBB, 1761-L20BWB-5A, 1761-L32BBB, and 1761-L32BWB controllers, the user supply 24V dc IN and chassis ground are internally connected.

You must also provide an acceptable grounding path for each device in your application. For more information on proper grounding guidelines, see the *Industrial Automation Wiring and Grounding Guidelines*, publication 1770-4.1.



ATTENTION: Remove the protective wrap before applying power to the controller. Failure to remove the wrap may cause the controller to overheat.

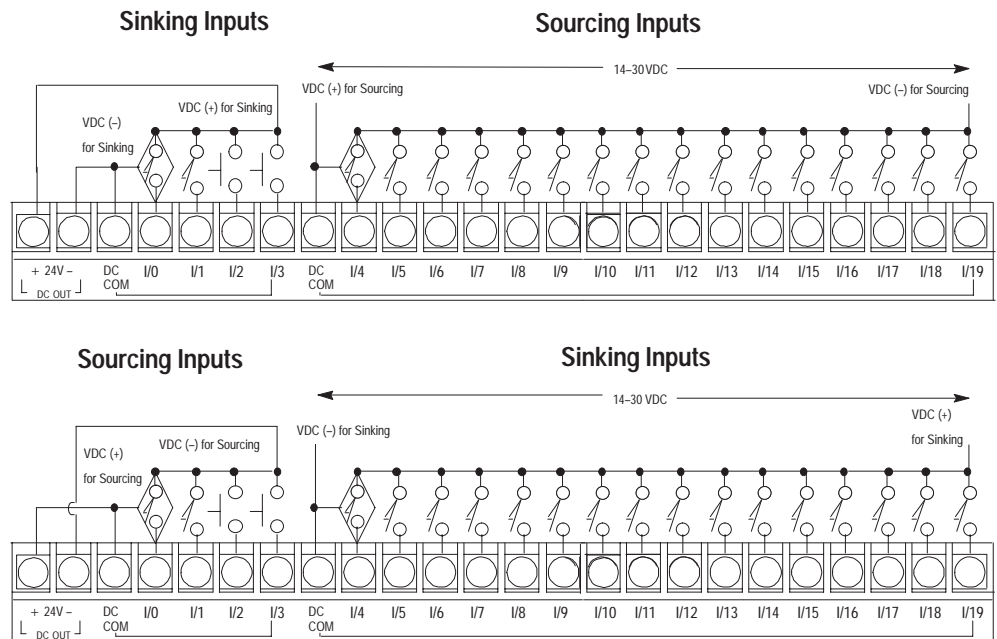
Sinking and Sourcing Circuits

Any MicroLogix 1000 DC inputs can be configured as sinking or sourcing depending on how the DC COM terminal is wired.

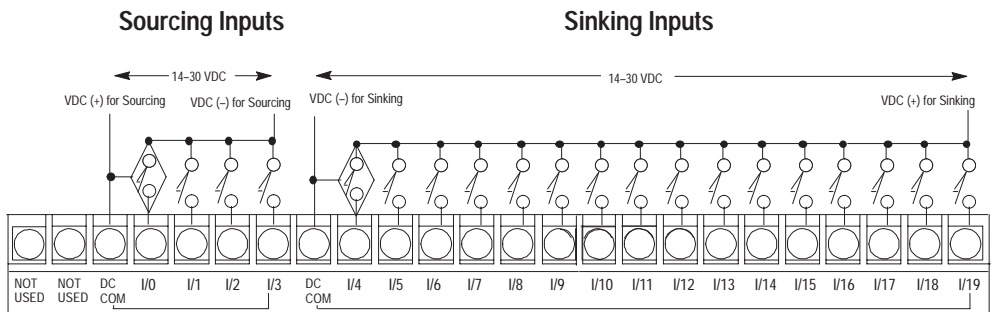
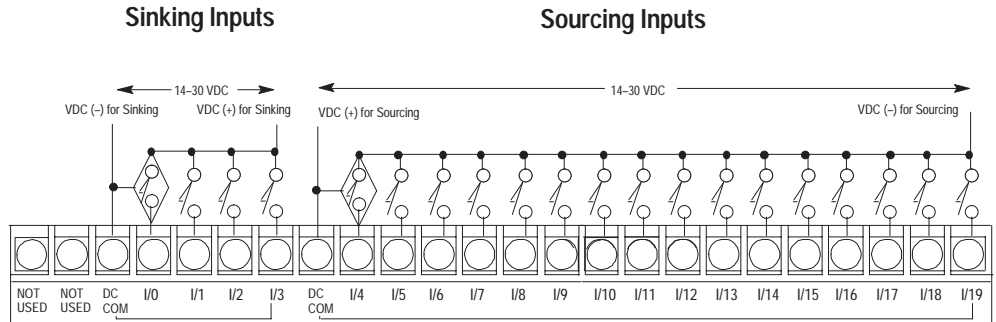
| Mode: | Definition: |
|----------|---|
| Sinking | The input energizes when high-level voltage is applied to the input terminal (active high). Connect the power supply VDC (-) to the MicroLogix DC COM terminal. |
| Sourcing | The input energizes when low-level voltage is applied to the input terminal (active low). Connect the power supply VDC (+) to the MicroLogix DC COM terminal. |

Sinking and Sourcing Wiring Examples

1761-L32BWA (Wiring diagrams also apply to 1761-L20BWA-5A, -L16BWA, -L10BWA.)



1761-L32BWB, -L32BBB (Wiring Diagrams also apply to 1761-L20BWB-5A -L16BWB, -L10BWB, -L16BBB.)



Wiring Recommendations



ATTENTION: Before you install and wire any device, disconnect power to the controller system.

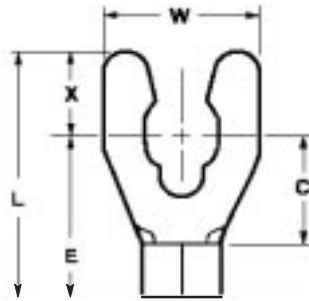
The following are general recommendations for wiring your controller system.

- Each wire terminal accepts 2 wires of the size listed below:

| Wire Type | Wire Size (2 wire maximum per terminal screw) |
|-----------|---|
| Solid | #14 to #22 AWG |
| Stranded | #16 to #22 AWG |

Refer to page 2–23 for wiring your high-speed counter.

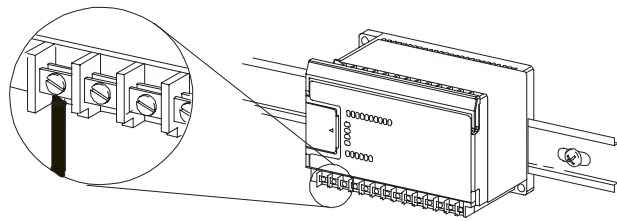
Important: The diameter of the terminal screw heads is 5.5 mm (0.220 in.). The input and output terminals of the micro controller are designed for the following spade lugs:



| Call-out | Dimension |
|----------|------------------------------|
| C | 6.35 mm (0.250 in.) |
| E | 10.95 mm (0.431 in.) maximum |
| L | 14.63 mm (0.576 in.) maximum |
| W | 6.35 mm (0.250 in.) |
| X | 3.56 mm (0.140 in.) |
| C+X | 9.91 mm (0.390 in.) maximum |

We recommend using either of the following AMP spade lugs: part number 53120-1, if using 22–16 AWG, or part number 53123-1, if using 16–14 AWG.

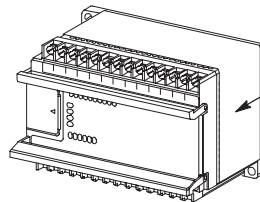
Important: If you use wires without lugs, make sure the wires are securely captured by the pressure plate. This is particularly important at the four end terminal positions where the pressure plate does not touch the outside wall.



20148I



ATTENTION: Be careful when stripping wires. Wire fragments that fall into the controller could cause damage. Do not strip wires above a mounted controller if the protective wrap is removed.



Protective
Wrap (remove after wiring)



ATTENTION: Remove the protective wrap before applying power to the controller. Failure to remove the wrap may cause the controller to overheat.



ATTENTION: Calculate the maximum possible current in each power and common wire. Observe all electrical codes dictating the maximum current allowable for each wire size. Current above the maximum ratings may cause wiring to overheat, which can cause damage.




ATTENTION: *United States Only:* If the controller is installed within a potentially hazardous environment, all wiring must comply with the requirements stated in the National Electrical Code 501-4 (b).

- Allow for at least 50 mm (2 in.) between I/O wiring ducts or terminal strips and the controller.
- Route incoming power to the controller by a path separate from the device wiring. Where paths must cross, their intersection should be perpendicular.
Important: Do not run signal or communications wiring and power wiring in the same conduit. Wires with different signal characteristics should be routed by separate paths.
- Separate wiring by signal type. Bundle wiring with similar electrical characteristics together.
- Separate input wiring from output wiring.
- Label wiring to all devices in the system. Use tape, shrink-tubing, or other dependable means for labeling purposes. In addition to labeling, use colored insulation to identify wiring based on signal characteristics. For example, you may use blue for dc wiring and red for ac wiring.

Wiring Diagrams, Discrete Input and Output Voltage Ranges

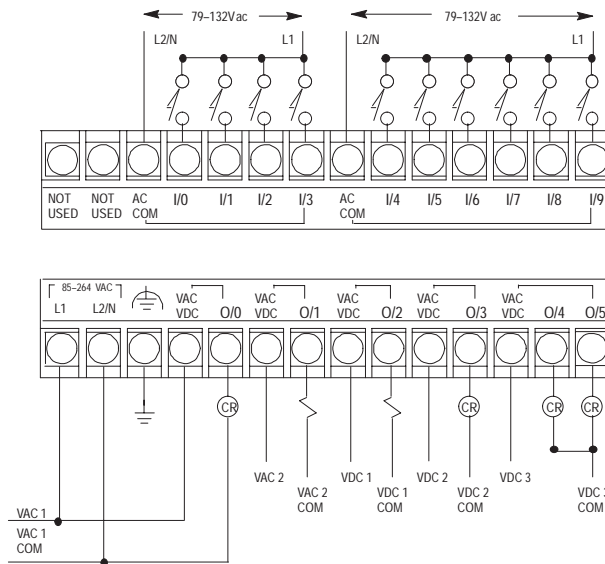
The following pages show the wiring diagrams, discrete input voltage ranges and discrete output voltage ranges. Controllers with dc inputs can be wired as either sinking or sourcing configurations. (Sinking and sourcing does not apply to ac inputs.)

Important:  This symbol denotes a functional earth ground terminal which provides a low impedance path between electrical circuits and earth for non-safety purposes, such as noise immunity improvement.



ATTENTION: The 24V dc sensor power source should not be used to power output circuits. It should only be used to power input devices (e.g. sensors, switches). Refer to page 1–3 for information on MCR wiring in output circuits.

1761-L16AWA Wiring Diagram



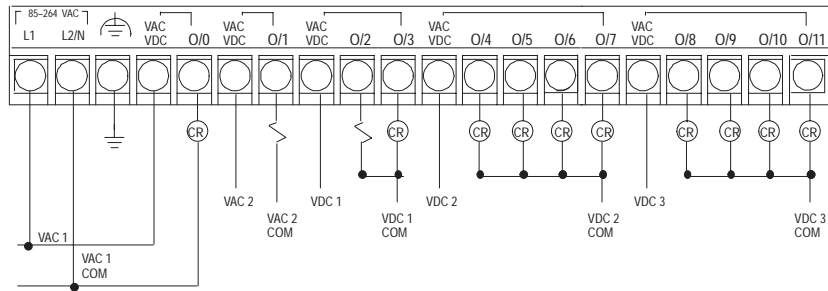
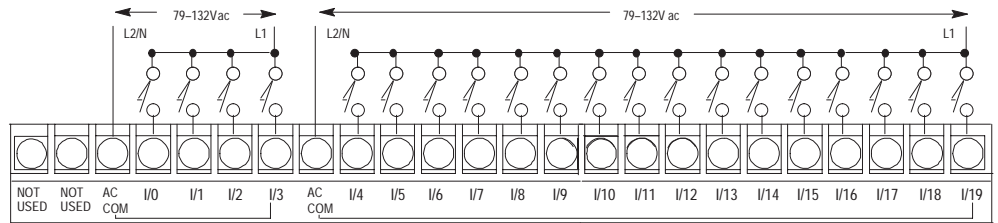
1761-L16AWA Input Voltage Range



1761-L16AWA Output Voltage Range



1761-L32AWA Wiring Diagram



1761-L32AWA Input Voltage Range

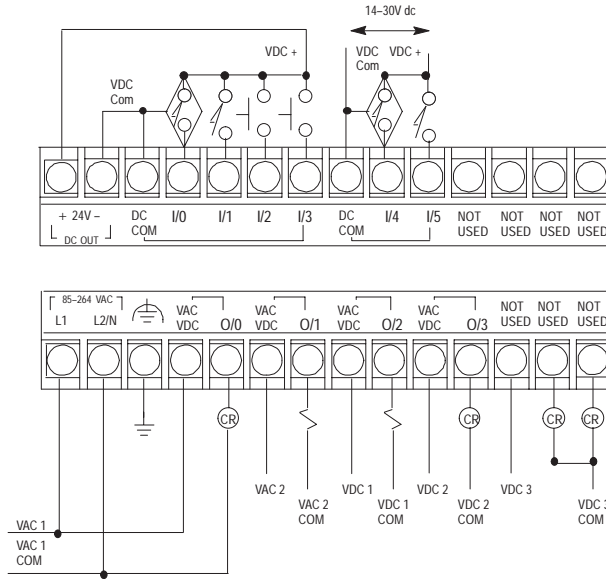


1761-L32AWA Output Voltage Range

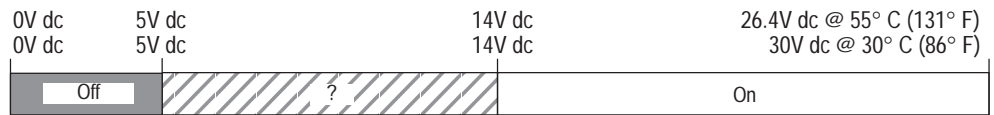


1761-L10BWA Wiring Diagram (Sinking Input Configuration)

Note: Refer to page 2-2 for additional input configuration options.



1761-L10BWA Input Voltage Range

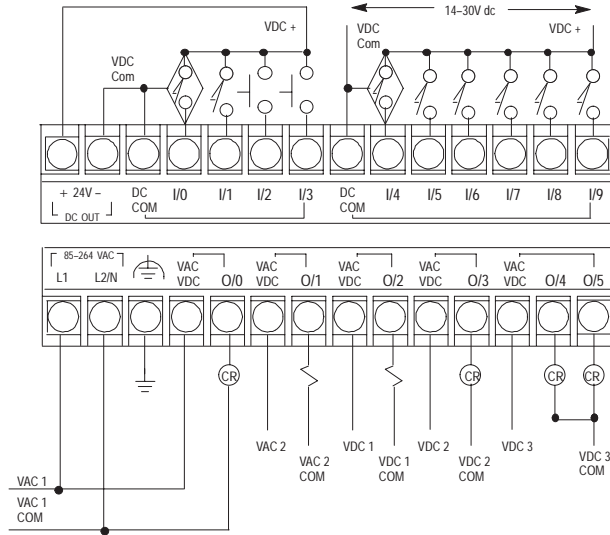


1761-L10BWA Output Voltage Range



1761-L16BWA Wiring Diagram (Sinking Input Configuration)

Note: Refer to page 2-2 for additional input configuration options.



1761-L16BWA Input Voltage Range

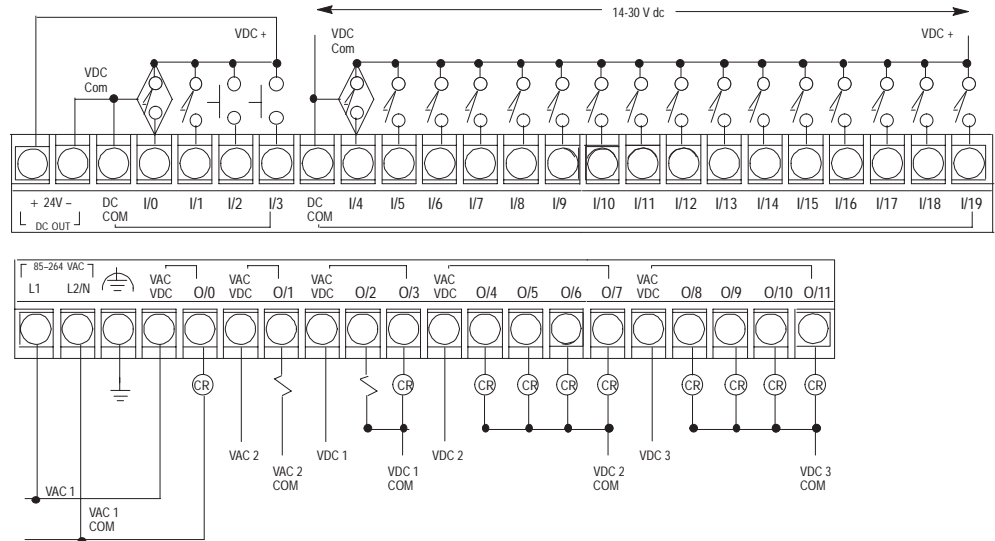
| | | | |
|-------|-------|--------|---------------------------|
| 0V dc | 5V dc | 14V dc | 26.4V dc @ 55° C (131° F) |
| 0V dc | 5V dc | 14V dc | 30V dc @ 30° C (86° F) |
| Off | ? | | On |

1761-L16BWA Output Voltage Range

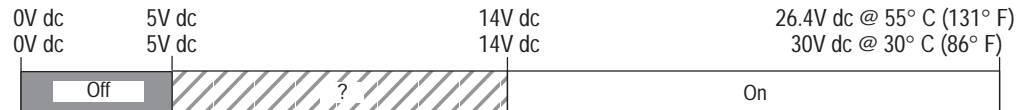
| | | |
|-------|-----------------|---------|
| 0V ac | 5V ac | 264V ac |
| 0V dc | 5V dc | 125V dc |
| ? | Operating Range | |

1761-L32BWA Wiring Diagram (Sinking Input Configuration)

Note: Refer to page 2-2 for additional input configuration options.



1761-L32BWA Input Voltage Range

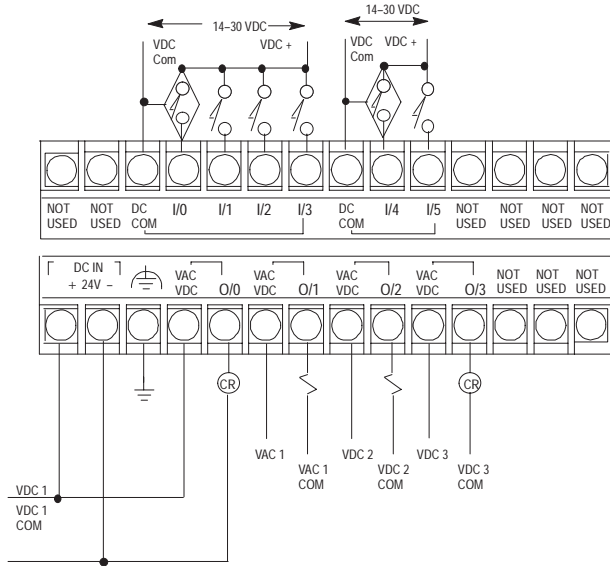


1761-L32BWA Output Voltage Range

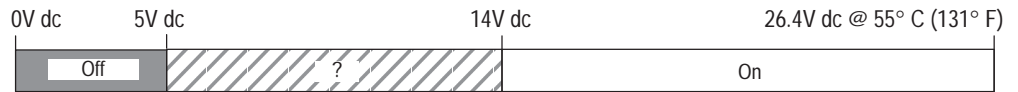


1761-L10BWB Wiring Diagram (Sinking Input Configuration)

Note: Refer to page 2-2 for additional input configuration options.



1761-L10BWB Input Voltage Range

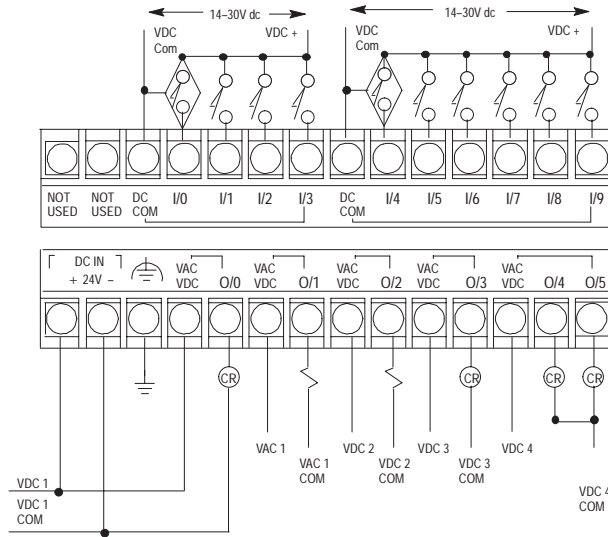


1761-L10BWB Output Voltage Range

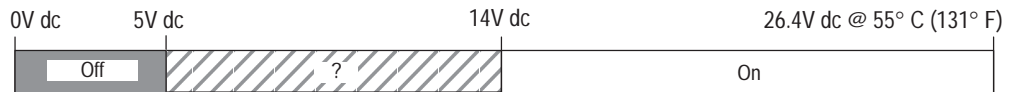


1761-L16BWB Wiring Diagram (Sinking Input Configuration)

Note: Refer to page 2-2 for additional input configuration options.



1761-L16BWB Input Voltage Range

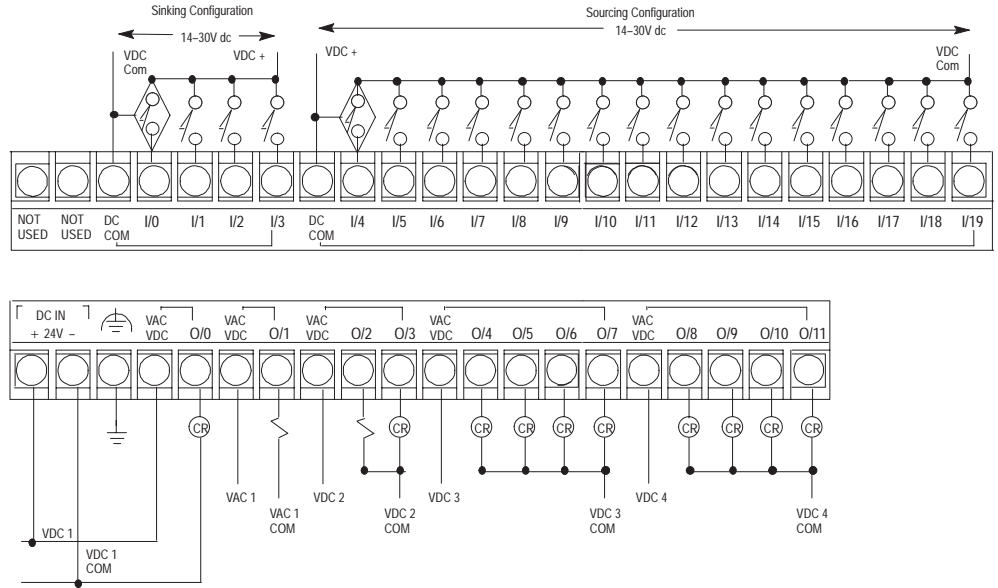


1761-L16BWB Output Voltage Range

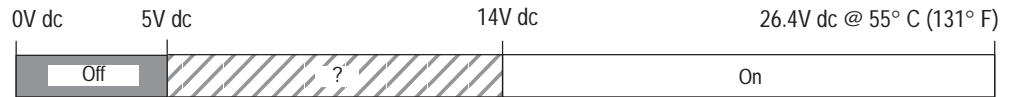


1761-L32BWB Wiring Diagram (Sinking Input Configuration)

Note: Refer to page 2-2 for additional input configuration options.



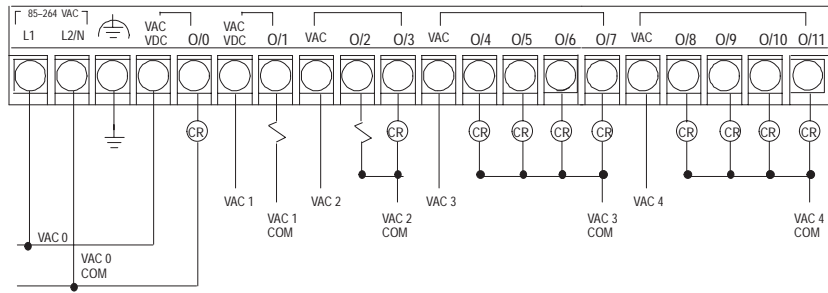
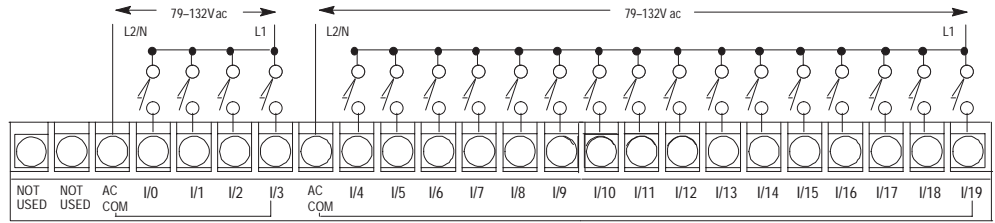
1761-L32BWB Input Voltage Range



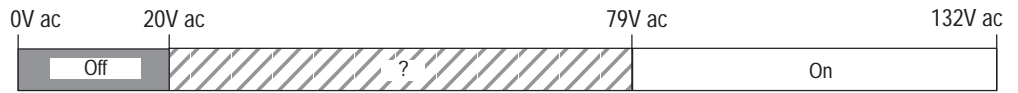
1761-L32BWB Output Voltage Range



1761-L32AAA Wiring Diagram



1761-L32AAA Input Voltage Range

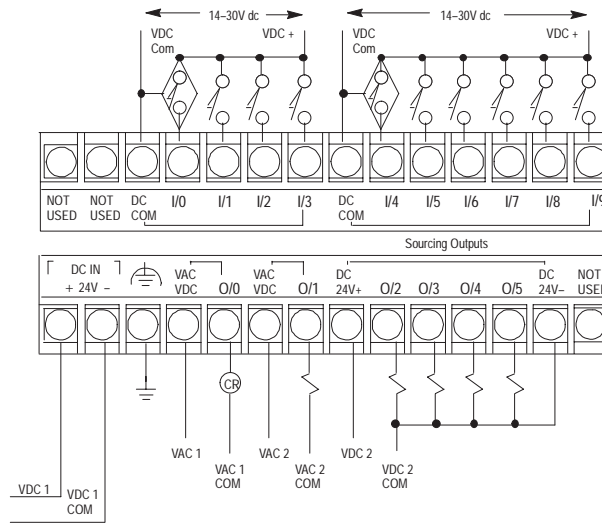


1761-L32AAA Output Voltage Range

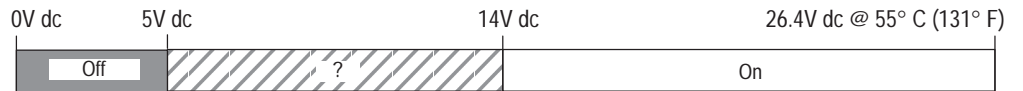


1761-L16BBB Wiring Diagram (Sinking Input Configuration)

Note: Refer to page 2-2 for additional input configuration options.



1761-L16BBB Input Voltage Range

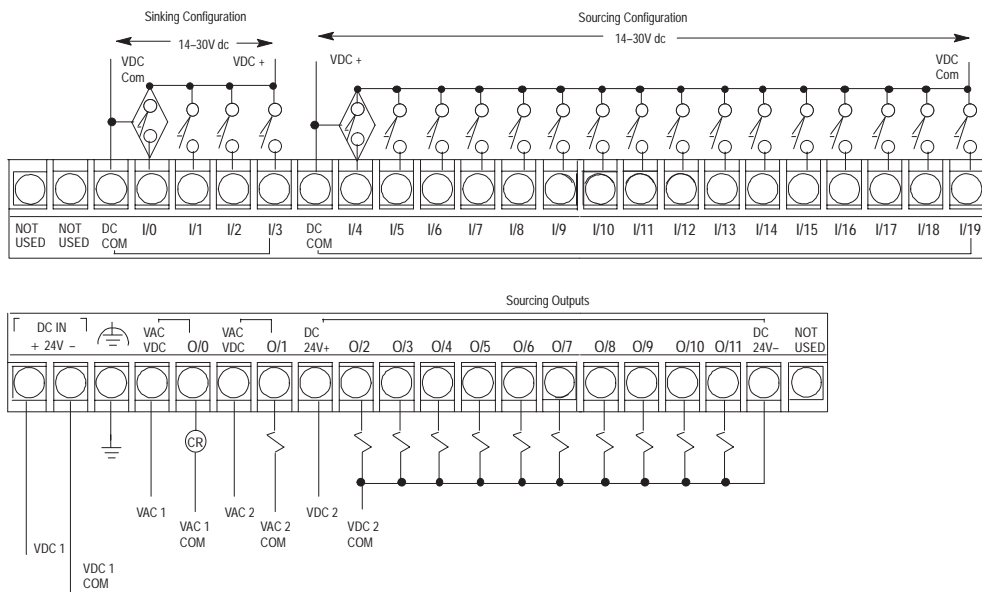


1761-L16BBB Output Voltage Range

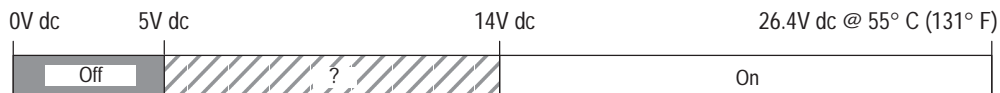


1761-L32BBB Wiring Diagram (Sinking Input Configuration)

Note: Refer to page 2-2 for additional input configuration options.



1761-L32BBB Input Voltage Range

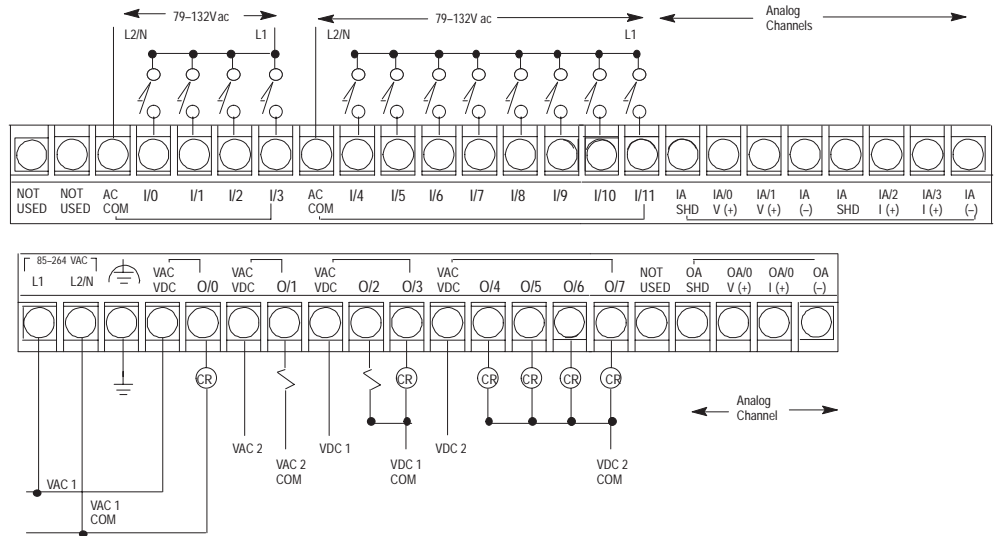


1761-L32BBB Output Voltage Range



1761-L20AWA-5A Wiring Diagram

Note: Refer to pages 2–20 through 2–22 for additional information on analog wiring.



1761-L20AWA-5A Discrete Input Voltage Range



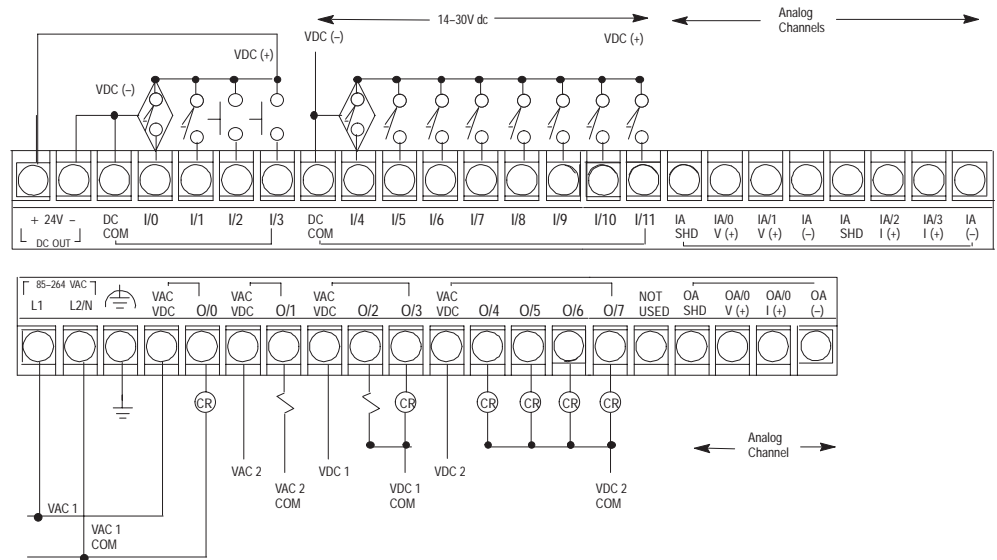
1761-L20AWA-5A Relay Output Voltage Range



1761-L20BWA-5A Wiring Diagram (Sinking Input Configuration)

Note: Refer to page 2–2 for additional discrete configuration options.

Refer to pages 2–20 through 2–22 for additional information on analog wiring.



1761-L20BWA-5A Discrete Input Voltage Range

| | | | |
|-------|-------|--------|---------------------------|
| 0V dc | 5V dc | 14V dc | 26.4V dc @ 55° C (131° F) |
| 0V dc | 5V dc | 14V dc | 30V dc @ 30° C (86° F) |
| Off | ? | ? | On |

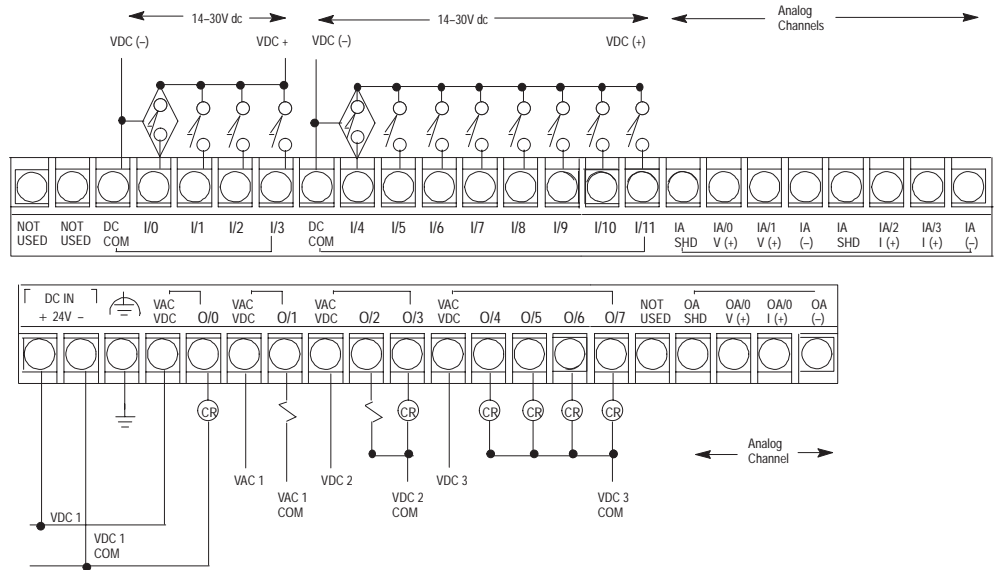
1761-L20BWA-5A Relay Output Voltage Range

| | | |
|-------|-----------------|---------|
| 0V ac | 5V ac | 264V ac |
| 0V dc | 5V dc | 125V dc |
| ? | Operating Range | |

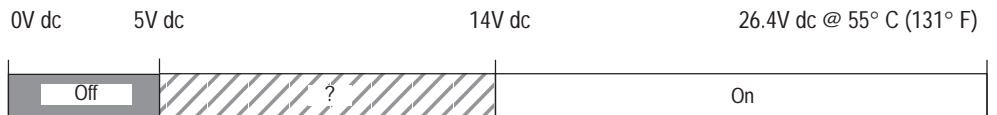
1761-L20BWB-5A Wiring Diagram (Sinking Input Configuration)

Note: Refer to page 2–2 for additional discrete configuration options.

Refer to pages 2–20 through 2–22 for additional information on analog wiring.



1761-L20BWB-5A Discrete Input Voltage Range



1761-L20BWB-5A Relay Output Voltage Range



Minimizing Electrical Noise on Analog Controllers

Inputs on analog employ digital high frequency filters that significantly reduce the effects of electrical noise on input signals. However, because of the variety of applications and environments where analog controllers are installed and operated, it is impossible to ensure that all environmental noise will be removed by the input filters.

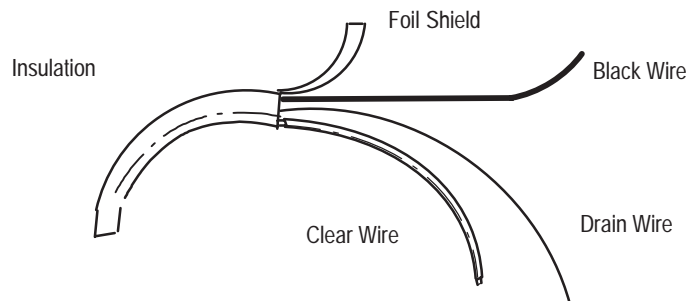
Several specific steps can be taken to help reduce the effects of environmental noise on analog signals:

- install the MicroLogix 1000 system in a properly rated (i.e., NEMA) enclosure. Make sure that the MicroLogix 1000 system is properly grounded.
- use Belden™ cable #8761 for wiring the analog channels making sure that the drain wire and foil shield are properly earth grounded.
- route the Belden cable separate from any other wiring. Additional noise immunity can be obtained by routing the cables in grounded conduit.

A system may malfunction due to a change in the operating environment after a period of time. We recommend periodically checking system operation, particularly when new machinery or other noise sources are installed near the MicroLogix 1000 system.

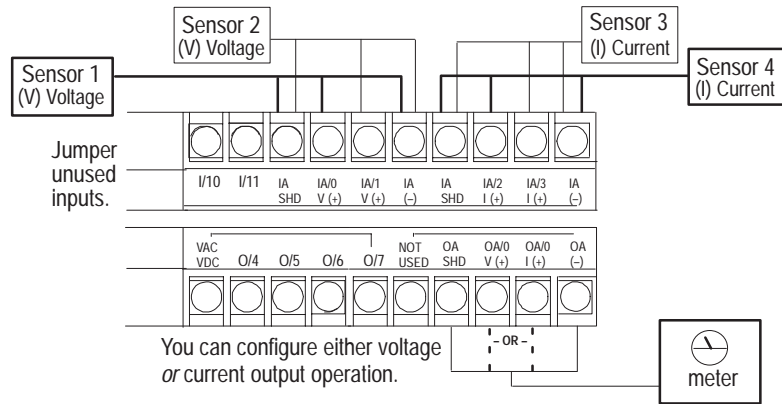
Grounding Your Analog Cable

Use shielded communication cable (Belden #8761). The Belden cable has two signal wires (black and clear), one drain wire and a foil shield. The drain wire and foil shield must be grounded at one end of the cable. *Do not* earth ground the drain wire and foil shield at *both* ends of the cable.



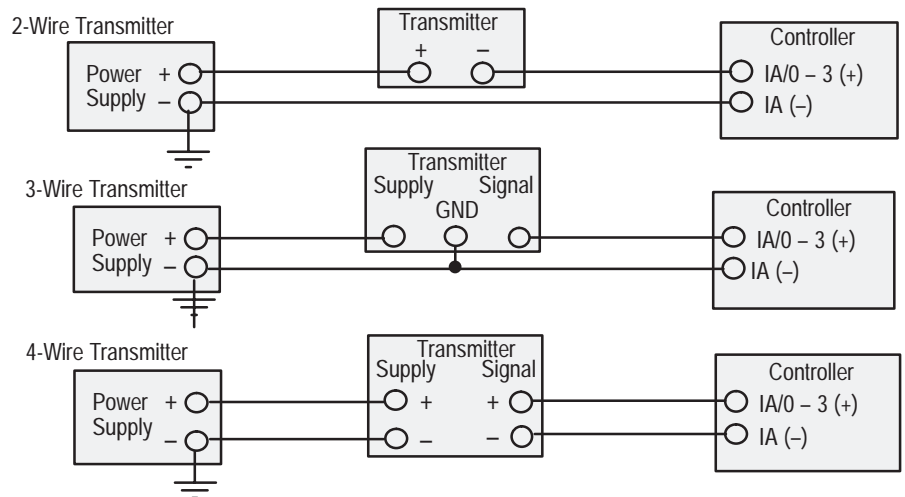
Wiring Your Analog Channels

Analog input circuits can monitor current *and* voltage signals and convert them to serial digital data. The analog output can support either a voltage *or* a current function.



For increased noise immunity, connect a ground wire directly from the shield terminals to chassis ground.

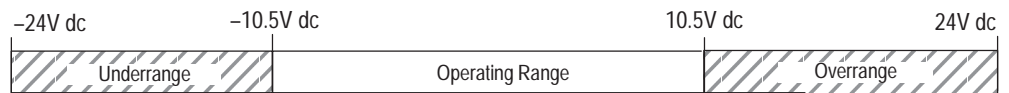
Important: The controller does *not* provide loop power for analog inputs. Use a power supply that matches the transmitter specifications.



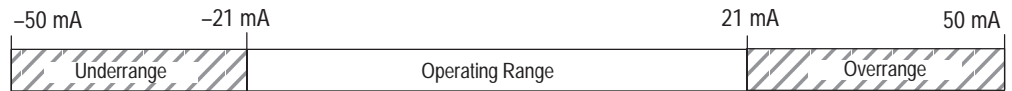
Analog Voltage and Current Input and Output Ranges

The following drawings show the analog voltage input range, analog current input range, analog voltage output range and analog current output range.

Analog Voltage Input Range



Analog Current Input Range



Note: The analog voltage inputs are protected to withstand the application of $\pm 24\text{V}$ dc without damage to the controller. The analog current inputs are protected to withstand the application of ± 50 mA without damage.

Analog Voltage Output Range



Analog Current Output Range



Note: The analog outputs are protected to withstand the short circuiting of the voltage or current outputs without damage to the controller.

For information on analog signal and data word values using the nominal transfer function formula, see page 7-4.

Wiring Your Controller for High-Speed Counter Applications

To wire the controller for high-speed counter applications, use input terminals I/0, I/1, I/2, and I/3. Refer to chapter 14 for information on using the high-speed counter.

Shielded cable is required for high-speed input signals 0–3 when the filter setting is set to either 0.10 ms or 0.075 ms. We recommend Belden #9503 for lengths up to 305 m (1000 ft). Shields should be grounded only at the signal source end of the cable. Ground the shield to the case of the signal source, so energy coupled to the shield is not delivered to the signal source's electronics.

Connecting the System

This chapter describes how to wire your controller system. The method you use and cabling required to connect your controller depends on what type of system you are employing. Specifically, this chapter contains information on:

- connecting the HHP
- DH-485 connections
- establishing communication

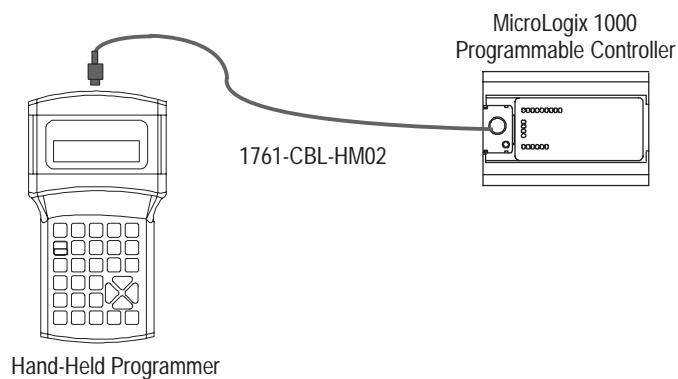
Important: In order to access the functionality of the Series C or later discrete and all MicroLogix 1000 analog controllers, you must configure your program to operate with these controllers. See page 18–18 for more information.

Connecting the HHP

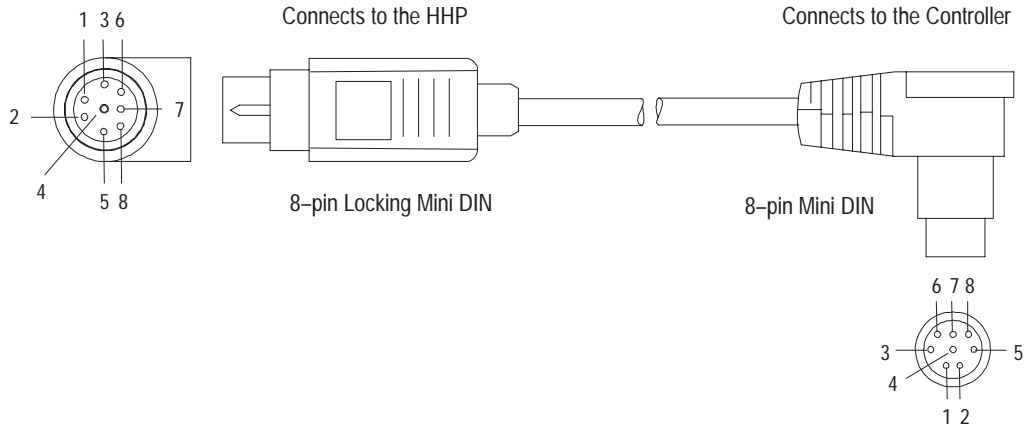
Use a serial cable to connect the MicroLogix 1000 HHPs RS-232 communication channel to the MicroLogix 1000 programmable controller, as shown below.



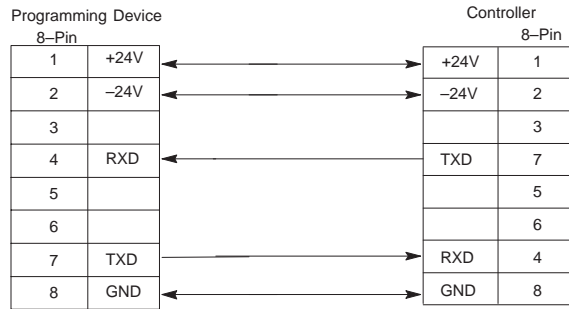
ATTENTION: Chassis ground, user 24V ground, and RS-232 ground are internally connected. You must connect the chassis ground terminal screw to chassis ground prior to connecting any devices. It is important that you understand your programming device's grounding system before connecting to the controller.



The 1761-CBL-HM02 Series B or higher cable with pinouts is shown below. Use this cable to connect the MicroLogix 1000 HHP to the MicroLogix 1000 Programmable Controller.

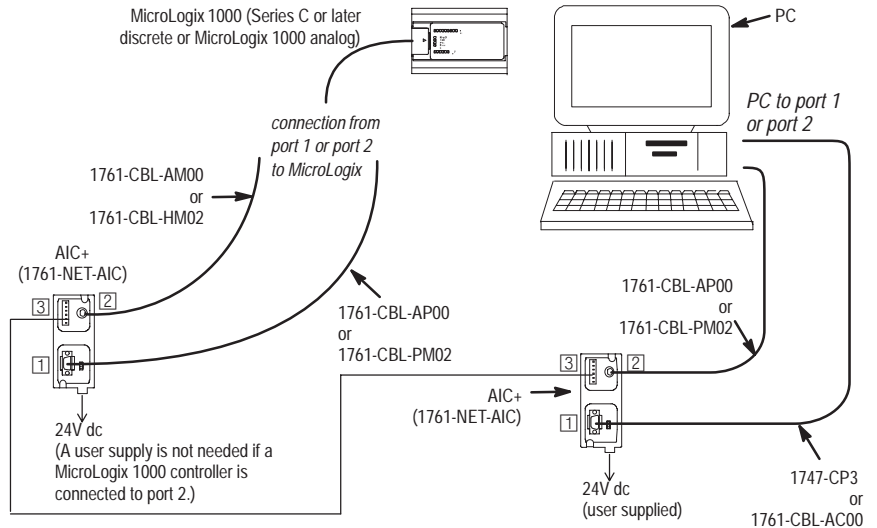


20188



Connecting to a DH-485 Network

Important: Only Series C or later MicroLogix 1000 discrete controllers and all MicroLogix 1000 analog controllers support DH-485 network connections. In order to access the DH-485 functionality of the Series C or later MicroLogix 1000 discrete and MicroLogix 1000 analog controllers, you must configure your program to operate with these controllers. See page 18–18 for more information.



MicroLogix DH-485 Network

- 1 DB-9 RS-232 port
- 2 mini-DIN 8 RS-232 port
- 3 DH-485 port

Recommended Tools

To connect a DH-485 network, you need tools to strip the shielded cable and to attach the cable and terminators to the AIC+ Advanced Interface Converter. We recommend the following equipment (or equivalent):

| Description | Part Number | Manufacturer |
|-----------------------------|-----------------|------------------|
| Shielded Twisted Pair Cable | #3106A or #9842 | Belden |
| Stripping Tool | 45-164 | Ideal Industries |
| 1/8 " Slotted Screwdriver | Not Applicable | Not Applicable |

DH-485 Communication Cable

The suggested DH-485 communication cable is either Belden #3106A or #9842. The cable is jacketed and shielded with one or two twisted wire pairs and a drain wire.

One pair provides a balanced signal line, and one additional wire is used for a common reference line between all nodes on the network. The shield reduces the effect of electrostatic noise from the industrial environment on network communication.

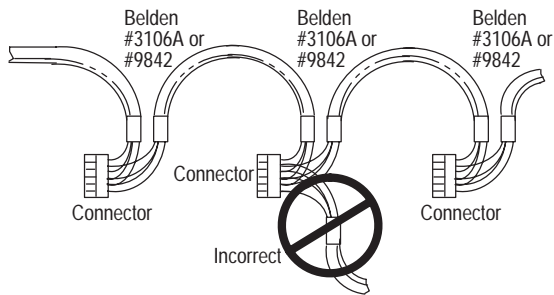
The communication cable consists of a number of cable segments daisy-chained together. The total length of the cable segments cannot exceed 1219 m (4000 ft).

When cutting cable segments, make them long enough to route them from one AIC+ to the next with sufficient slack to prevent strain on the connector. Allow enough extra cable to prevent chafing and kinking in the cable.

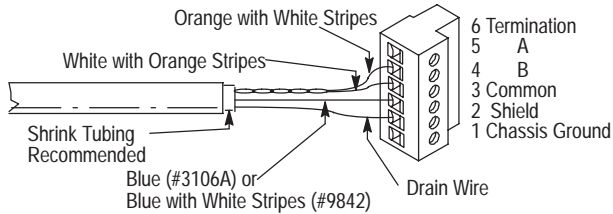
Use these instructions for wiring the Belden #3106A or #9842 cable. (If you are using standard Allen-Bradley cables, see the Cable Selection Guide on page 3–8.)

Connecting the Communication Cable to the DH-485 Connector

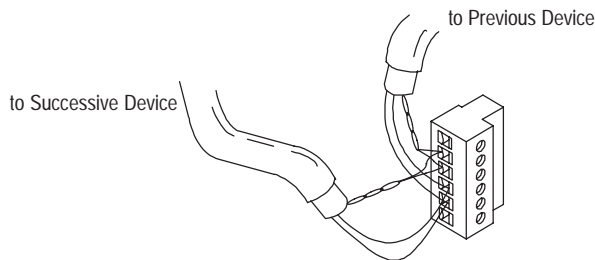
Important: A daisy-chained network is recommended. We do **not** recommend the following:



Single Cable Connection



Multiple Cable Connection



The table below shows connections for Belden #3106A.

| For this Wire/Pair | Connect this Wire | To this Terminal |
|--------------------|--------------------------|-----------------------|
| Shield/Drain | Non-jacketed | Terminal 2 – Shield |
| Blue | Blue | Terminal 3 – (Common) |
| White/Orange | White with Orange Stripe | Terminal 4 – (Data B) |
| | Orange with White Stripe | Terminal 5 – (Data A) |

The table below shows connections for Belden #9842.

| For this Wire/Pair | Connect this Wire | To this Terminal |
|--------------------|--------------------------|---------------------------------------|
| Shield/Drain | Non-jacketed | Terminal 2 – Shield |
| Blue/White | White with Blue Stripe | Cut back – no connection ^① |
| | Blue with White Stripe | Terminal 3 – (Common) |
| White/Orange | White with Orange Stripe | Terminal 4 – (Data B) |
| | Orange with White Stripe | Terminal 5 – (Data A) |

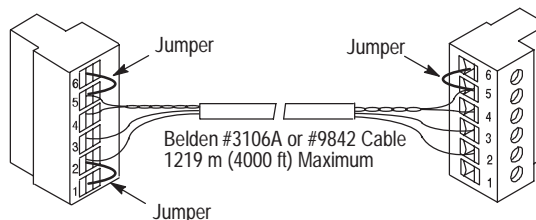
^① To prevent confusion when installing the communication cable, cut back the white with blue stripe wire immediately after the the insulation jacket is removed. This wire is not used by DH-485.

Grounding and Terminating the DH-485 Network

Only one connector at the end of the link must have Terminals 1 and 2 jumpered together. This provides an earth ground connection for the shield of the communication cable.

Both ends of the network must have Terminals 5 and 6 jumpered together. This connects the termination impedance (of 120Ω) that is built into each AIC+ as required by the DH-485 specification.

End-of-Line Termination



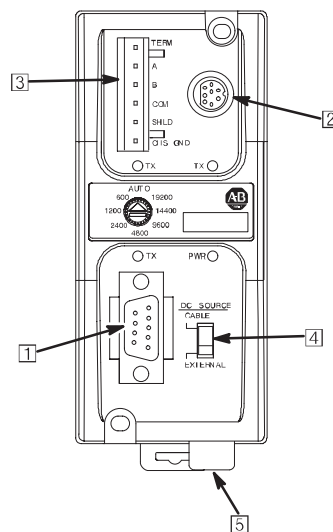
Connecting the AIC+

Important: Only Series C or later MicroLogix 1000 discrete controllers and all MicroLogix 1000 analog controllers support DH-485 network connections.

You can connect an unpowered AIC+, catalog number 1761-NET-AIC, to the network without disrupting network activity. In addition, if a MicroLogix 1000 controller powers an AIC+ that is connected to the network, network activity will not be disrupted should the MicroLogix 1000 controller be removed from the AIC+.

The figure below shows the external wiring connections and specifications of the AIC+.

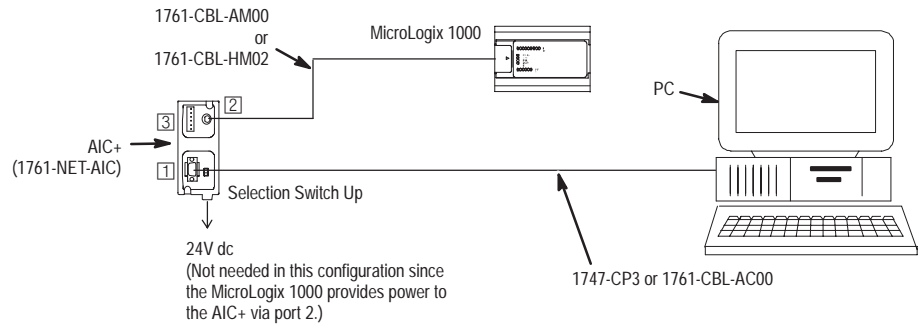
**AIC+ Advanced Interface Converter
(1761-NET-AIC)**



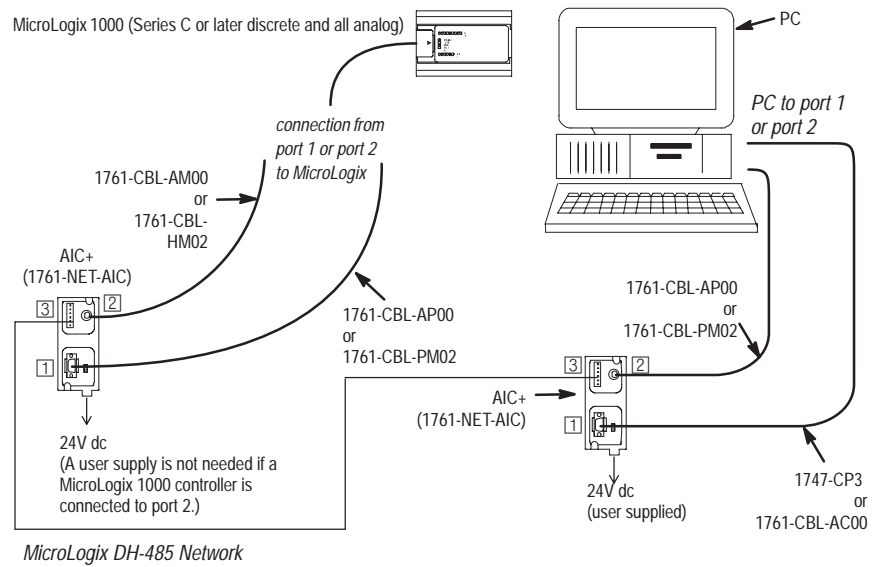
| Item | Description |
|------|--|
| 1 | Port 1 – DB-9 RS-232, DTE |
| 2 | Port 2 – mini-DIN 8 RS-232 |
| 3 | Port 3 – DH-485 Phoenix plug |
| 4 | DC Power Source selector switch (cable = port 2 power source, external = external power source connected to item 5) |
| 5 | Terminals for external 24V dc power supply and chassis ground |

For additional information on connecting to the AIC+, see the *Advanced Interface Converter (AIC+) and DeviceNet Interface (DNI) Installation Instructions*, Publication 1761-5.11.

DF1 Isolated Point-to-Point Connection

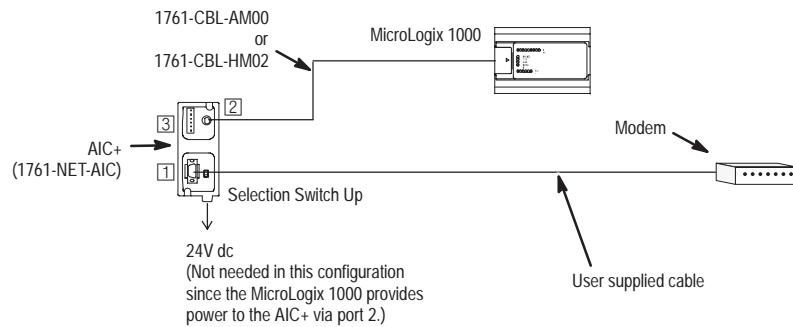


DH-485 Network Connection



- 1 DB-9 RS-232 port
- 2 mini-DIN 8 RS-232 port
- 3 DH-485 port

DF1 Isolated Modem Connection



Cable Selection Guide



| Cable | Length | Connections from | to AIC+ | External Power Supply Required ^① | Power Selection Switch Setting ^① |
|---------------|-----------------|---|---------|---|---|
| 1747-CP3 | 3m (9.8 ft) | SLC 5/03 or SLC 5/04 processor, channel 0 | port 1 | yes | external |
| 1761-CBL-AC00 | 45 cm (17.7 in) | PC COM port | port 1 | yes | external |
| | | PanelView 550 through NULL modem adapter | port 1 | yes | external |
| | | Port 1 on another AIC+ | port 1 | yes | external |



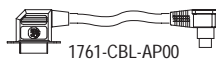
| Cable | Length | Connections from | to AIC+ | External Power Supply Required ^① | Power Selection Switch Setting ^① |
|---------------|-----------------|---|---------|---|---|
| 1761-CBL-AS03 | 3m (9.8 ft) | SLC 500 Fixed, | port 3 | yes | external |
| 1761-CBL-AS09 | 9.5m (31.17 ft) | SLC 5/01, SLC 5/02, and SLC 5/03 processors | | | |
| | | PanelView 550 RJ45 port | port 3 | yes | external |



| Cable | Length | Connections from | to AIC+ | External Power Supply Required | Power Selection Switch Setting |
|----------------------------|-----------------|---------------------------|---------|--------------------------------|--------------------------------|
| 1761-CBL-AM00 | 45 cm (17.7 in) | MicroLogix 1000 | port 2 | no | cable |
| 1761-CBL-HM02 ^② | 2m (6.5 ft) | to port 2 on another AIC+ | port 2 | yes | external |

① External power supply required unless the AIC+ is powered by the device connected to port 2, then the selection switch should be set to cable.

② Series B cables or higher are required for hardware handshaking.



| Cable | Length | Connections from | to AIC+ | External Power Supply Required | Power Selection Switch Setting |
|----------------------------|-----------------|--|---------|--------------------------------|--------------------------------|
| 1761-CBL-AP00 | 45 cm (17.7 in) | SLC 5/03 or SLC 5/04 processors, channel 0 | port 2 | yes | external |
| 1761-CBL-PM02 ^② | 2m (6.5 ft) | MicroLogix 1000 | port 1 | yes ^① | external ^① |
| | | PanelView 550 through NULL modem adapter | port 2 | yes | external |
| | | PC COM port | port 2 | yes | external |



| Cable | Length | Connections from | to AIC+ | External Power Supply Required | Power Selection Switch Setting |
|-------------------|--------|-------------------------------------|---------|--------------------------------|--------------------------------|
| straight 9-25 pin | -- | modem or other communication device | port 1 | yes ^① | external ^① |

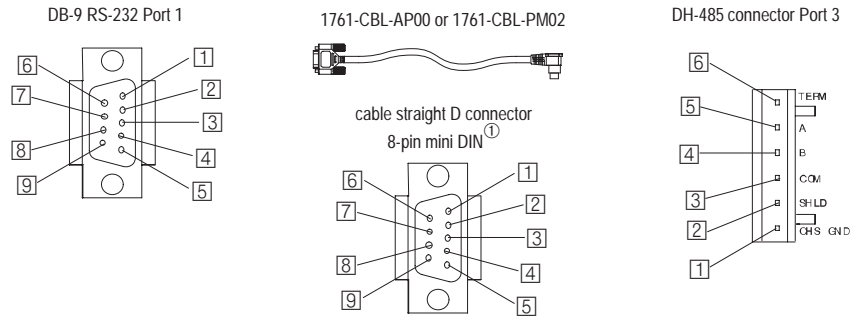
^① External power supply required unless the AIC+ is powered by the device connected to port 2, then the selection switch should be set to cable.

^② Series B cables or higher are required for hardware handshaking.

Recommended User-Supplied Components

These components can be purchased from your local electronics supplier.

| Component | Recommended Model |
|--|---|
| external power supply and chassis ground | power supply rated for 20.4–28.8V dc |
| NULL modem adapter | standard AT |
| straight 9–25 pin RS-232 cable | see table below for port information if making own cables |



| Item | Port 1 DB-9 RS-232 | Port 2 ^① (1761-CBL-PM02 cable) | Port 3 DH-485 Connector |
|------|-------------------------------------|--|----------------------------|
| ① | received line signal detector (DCD) | same state as port 1's DCD signal | chassis ground |
| ② | received data (RxD) | received data (RxD) | cable shield |
| ③ | transmitted data (TxD) | transmitted data (TxD) | signal ground |
| ④ | DTE ready (DTR) | DTE ready (DTR) | DH-485 data B |
| ⑤ | signal common (GRD) | signal common (GRD) | DH-485 data A |
| ⑥ | DCE ready (DSR) | DCE ready (DSR) | termination |
| ⑦ | request to send (RTS) | request to send (RTS) | not applicable |
| ⑧ | clear to send (CTS) | clear to send (CTS) | not applicable |
| ⑨ | not applicable | not applicable | not applicable |

^① An 8-pin mini DIN connector is used for making connections to port 2. This connector is not commercially available. If you are making a cable to connect to port 2, you must configure your cable to connect to the Allen-Bradley cable shown above.

^② On port 1, pin 4 is electronically jumpered to pin 6. Whenever the AIC+ is powered on, pin 4 will match the state of pin 6.

^③ In the 1761-CBL-PM02 cable, pins 4 and 6 are jumpered together within the DB-9 connector.

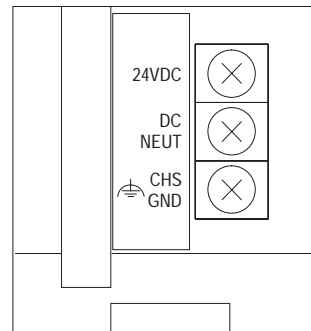
Powering the AIC+



ATTENTION: If you use an external power supply, it must be 24V dc. Permanent damage will result if miswired with the wrong power source.

Set the DC Power Source selector switch to **EXTERNAL** before connecting the power supply to the AIC+.

Bottom View



ATTENTION: Always connect the CHS GND (chassis ground) terminal to the nearest earth ground. This connection must be made whether or not an external 24V dc supply is used.

In normal operation with the MicroLogix 1000 programmable controller connected to port 2 of the AIC+, the controller powers the AIC+. Any AIC+ not connected to a controller requires a 24V dc power supply. The AIC+ requires 85 mA at 24V dc.

If both the controller and external power are connected to the AIC+, the power selection switch determines what device powers the AIC+.

Power Options

Below are two options for powering the AIC+:

- Use the 24V dc user power supply (200 mA maximum) built into the MicroLogix controller. The AIC+ is powered through a hard-wired connection using a communication cable (1761-CBL-HM02, or equivalent) connected to port 2.
- Use an external DC power supply with the following specifications:
 - operating voltage: 24V dc +20% / -15%
 - output current: 120 mA minimum
 - rated NEC

Make a hard-wired connection from the external supply to the screw terminals on the bottom of the AIC+.



ATTENTION: If you use an external power supply, it must be 24V dc. Permanent damage will result if miswired with the wrong power source.

Installing and Attaching the AIC+

1. Take care when installing the AIC+ in an enclosure so that the cable connecting the MicroLogix 1000 controller to the AIC+ does not interfere with the enclosure door.
2. Carefully plug the terminal block into the DH-485 port on the AIC+ you are putting on the network. Allow enough cable slack to prevent stress on the plug.
3. Provide strain relief for the Belden cable after it is wired to the terminal block. This guards against breakage of the Belden cable wires.

Establishing Communication

When you connect a MicroLogix 1000 controller, it *automatically* determines which protocol is active (DF1 or DH-485), and establishes communication accordingly. Therefore, no special configuration is required to connect to either network.

However, to shorten the connection time, you can specify which protocol the controller should attempt to establish communication with first. This is done using the Primary Protocol bit, S:0/10. The default setting for this bit is DF1 (0). If the primary protocol bit is set to DF1, the MicroLogix 1000 controller will attempt to connect using the configured DF1 protocol: either full-duplex or half-duplex slave. To have the controller first attempt DH-485 communication, set this bit to 1.

For DH-485 networks that will only contain MicroLogix controllers, at least one controller must have its primary protocol bit set to 1 so that the network can be initialized.

Automatic Protocol Switching

The MicroLogix 1000 Series D or later discrete and all MicroLogix 1000 analog controllers perform automatic protocol switching between DH-485 and the configured DF1 protocol. (The controller cannot automatically switch between DF1 full-duplex and DF1 half-duplex slave.) This feature allows you to switch from active communication on a DF1 half-duplex network to the DH-485 protocol to make program changes.

Simply disconnect the MicroLogix controller from the DF1 half-duplex network and connect it to your personal computer. The controller recognizes the computer is attempting to communicate using the DH-485 protocol and automatically switches to it. When your program changes are complete, you can disconnect your computer, reconnect the modem, and the controller automatically switches back to the configured DF1 protocol. For example, if you are using the DH-485 protocol to make program changes and you connect an HHP, you can switch to active communication on a DF1 full-duplex network.

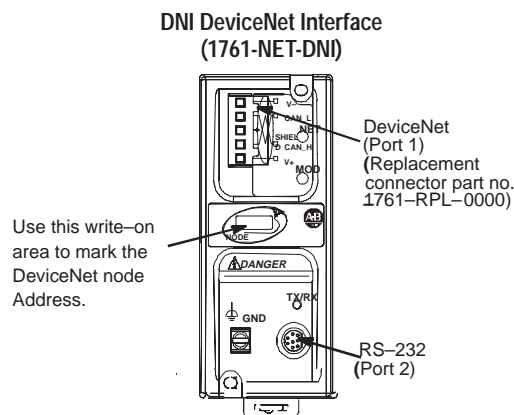
The following baud rate limitations affect autoswitching:

- If the configured DH-485 baud rate is 19200, the configured DF1 baud rate must be 4800 or greater.
- If the configured DH-485 baud rate is 9600, the configured DF1 baud rate must be 2400 or greater.

DeviceNet Communications

You can also connect a MicroLogix to a DeviceNet network using the DeviceNet Interface (DNI), catalog number 1761-NET-DNI. For additional information on connecting the DNI, see the *Advanced Interface Converter (AIC+) and DeviceNet Interface (DNI) Installation Instructions*, Publication 1761-5.11. For information on how to configure and commission a DNI, see the *DeviceNet Interface User Manual*, Publication 1761-6.5.

The figure that follows identifies the ports of the DNI.



Cable Selection Guide



| Cable | Length | Connections from | to DNI |
|----------------------------|-----------------|------------------------------|--------|
| 1761-CBL-AM00 | 45 cm (17.7 in) | MicroLogix 1000 (all series) | port 2 |
| 1761-CBL-HM02 ^① | 2m (6.5 ft) | MicroLogix 1000 (all series) | port 2 |



| Cable | Length | Connections from | to DNI |
|----------------------------|-----------------|--|--------|
| 1761-CBL-APM00 | 45 cm (17.7 in) | SLC 5/03 or SLC 5/04 processors, channel 0 | port 2 |
| 1761-CBL-PM02 ^① | 2m (6.5 ft) | PC COM port | port 2 |

^① Series B cables or higher are required for hardware handshaking.

Using Your Hand-Held Programmer

This chapter describes your MicroLogix 1000 Hand-Held Programmer (HHP), its memory module, and its power-up procedure. It also walks you through the start-up displays and helps you understand some of the functionality options available to you.

Read this chapter for information about:

- your HHP
- installing memory modules
- the keys you use
- the power-up sequence
- the HHPs functional areas
- the HHPs defaults

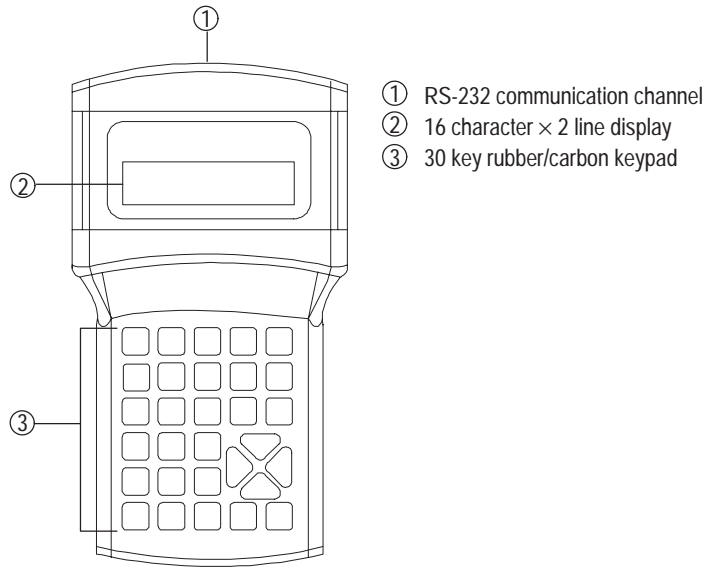
Important: The table below provides software compatibility information necessary for full functionality of your MicroLogix 1000 controller.

| Software Package | Version x.x | Functionality Level |
|---|-------------------|--|
| RSLogix 500 9324-RL03000END RSLogix 500 Starter 9323-RL0100END | v2.10.11 or later | Full functionality for MicroLogix 1000 Series D and analog controllers |
| | v2.0.57 | Full functionality for Series D controllers only |
| | v1.24 | Series C functionality only |
| | v1.05 | |
| A.I. 500 9323-S5300D A.I. Micro 9323-MX300EN | v8.16 or later | Full functionality for MicroLogix 1000 Series D and analog controllers |
| | v8.15 | Series C functionality only |
| | v8.14 or earlier | Series C functionality only; Cannot download to Series D or analog controllers |
| APS 9323-PA2E | v6.04 or earlier | Series A/B functionality only |
| MPS 9323-PA1E | v1.0 | |

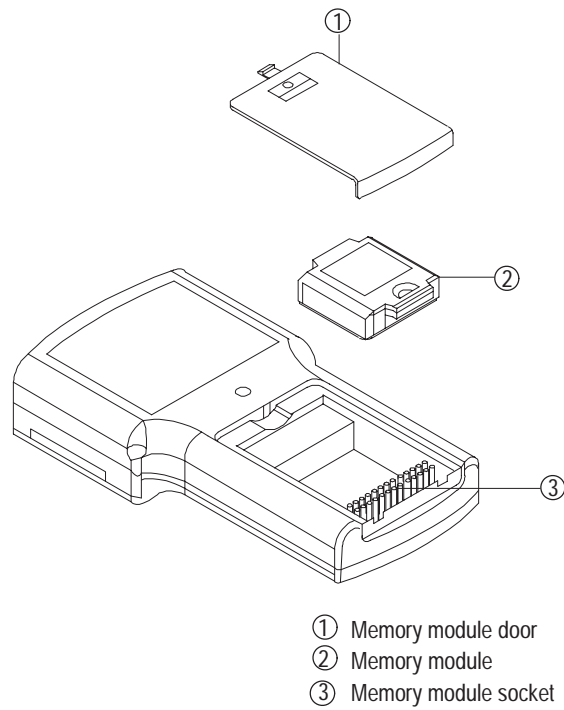
About Your HHP

The MicroLogix 1000 Hand-Held Programmer (HHP) allows you to create, edit, monitor, and troubleshoot Instruction List (Boolean) programs for your micro controller. With the HHP and either a 10-, 16-, 32-I/O point or analog micro controller, you eliminate the need for hard-wired relay logic. This device also allows you to transfer programs to and from an optional removable memory module.

The hardware features of the HHP are:



Additional hardware features of the MicroLogix 1000 HHP are:



Installing the Optional Memory Module

Two optional memory modules are available for the MicroLogix 1000 HHP:

- 8 Kbyte memory module, 1761-HHM-K08 – stores 1 program (possibly more than 1, depending on program size)
- 64 Kbyte memory module, 1761-HHM-K64 – stores a minimum of 8 programs

For information on loading and storing programs to your memory module, see page 19–1.



ATTENTION: Always remove power from the HHP before inserting or removing the memory module. This guards against possible damage to the module, as well as undesired controller faults.



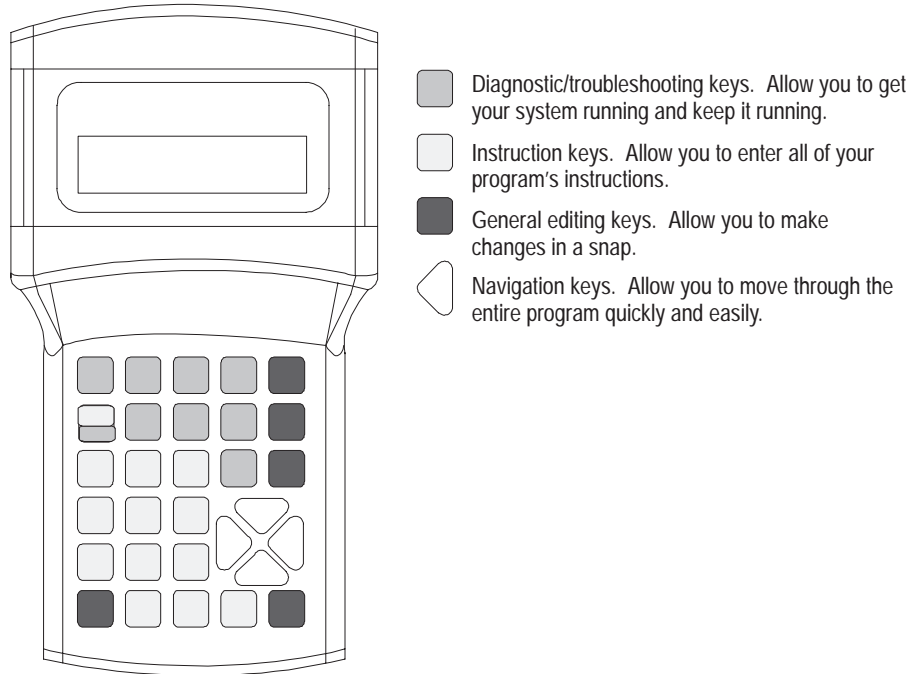
ATTENTION: To avoid potential damage to the memory modules, handle them by the ends of the carrier or edges of the plastic housing. Skin oil and dirt can corrode metallic surfaces, inhibiting electrical contact. Also, do not expose memory modules to surfaces or areas that may typically hold an electrostatic charge. Electrostatic charges can alter or destroy memory.

To insert a memory module use the following procedure:

1. If the MicroLogix 1000 HHP is connected to the controller, remove the cable from the HHP or turn off power to the controller.
2. Remove the memory module door.
3. Locate the socket on the processor board. Place the memory module onto the socket and press firmly in place.
4. Replace the memory module door.
5. If the MicroLogix 1000 HHP was connected to the controller, reconnect the cable to the HHP, or restore power to the controller.

The Keys You Use

When using the MicroLogix 1000 HHP, you will be pressing individual keys and key sequences for the purposes identified in the illustration below. Details about individual key functions and key sequences are provided in this manual at their point of use.

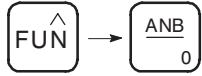
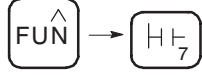
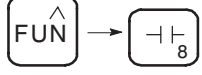
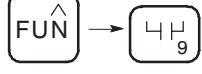
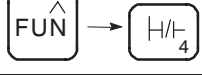

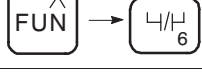


Understanding the Keys' Context Sensitivity

If you look at the labels on each key, you will notice that most of the keys perform more than one function. The MicroLogix 1000 HHP is designed to distinguish which function you want to perform, based on the context you are in at the time you press the key.

Accessing Additional Characters

Several characters are available that are not displayed on the keypad. These are outlined in the table below.

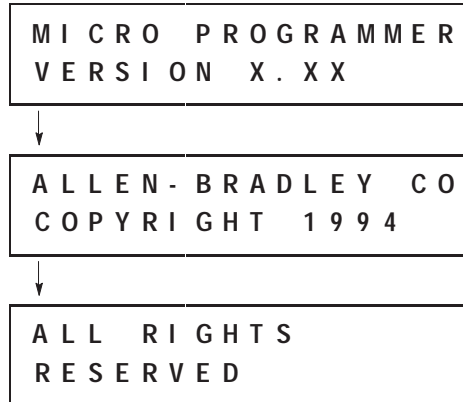
| To Access This Character: | Press This Key Sequence: |
|---------------------------|--|
| # |  |
| A |  |
| B |  |
| C |  |
| D |  |
| E |  |
| F |  |

These characters are useful for entering indexed addresses, hexadecimal values, and program names.

Identifying the Power-Up Sequence

When the MicroLogix 1000 HHP is first connected to the controller, the following sequence occurs:

1. The HHP performs diagnostic self tests. While doing this it displays the following Copyright screens:



These screens will always appear in English, even if you later select an alternate language for the HHP.

2. The HHP begins connecting to the controller and displays:



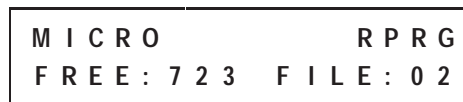
The HHP automatically finds the controller's baud rate and connects to it at that rate.

3. After a successful connection, the HHP displays the home screen.

For discrete controllers:



For analog controllers:



Important: If an error occurs during the power-up sequence, refer to chapter 20, *Troubleshooting*, for a list of error codes.

Understanding the HHPs Functional Areas

There are six main functional areas of the MicroLogix 1000 HHP, each with a unique purpose. They are:



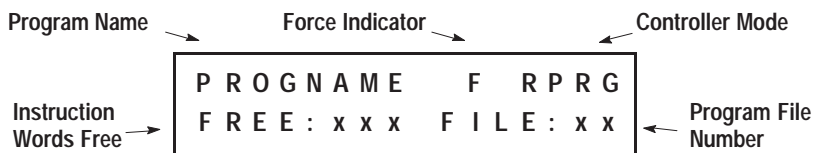
Descriptions of each of these areas and the tasks you can complete follow.

Home

Home is the functional area you enter after the HHP powers up. It provides important program and controller information. You can access all other functional areas from home.

Screen Definition

The following figure shows the home screen and identifies its main sections.



| Section | Description |
|------------------------|---|
| Program Name | The name of the program currently in the controller. |
| Force Indicator | If forces exist in the controller, an \mathbb{F} appears. If no forces exist, nothing appears. (See page 18–35 for information on forcing I/O.) |
| Controller Mode | The current mode of the controller is displayed. If program edits exist, the mode flashes. (See page 18–21 for information on valid modes.) |
| Instruction Words Free | The number of instruction words still available in the current program. |
| Program File Number | On entry to the program monitor, the file number to monitor defaults to this file. |



From Home you can access these areas:
Menu
Mode
Program Monitor
Data Monitor
Multi-Point Function

How to Complete Tasks

You complete tasks by pressing the appropriate key or key sequence from the home screen.

| To: | Press: |
|---|--------|
| access the menu options | |
| change the controller's mode (See page 18-23.) | |
| access the multi-point functional area (See page 18-31.) | |
| view faults (See page 20-11.) | |
| clear a fault manually (See page 20-11.) | → |
| access the program monitor functional area (See page 17-1.) | → |
| access the data monitor functional area (See page 18-27.) | → → |

Access Menu by pressing this key:



Menu

From the menu functional area, you can perform various program and system tasks. The available menu options are:

1. **LANGUAGE** (See page 4-17.)
2. **ACCEPT EDITS** (See page 18-21.)
3. **PROG CONFIG** (See page 18-1.)
4. **MEM MODULE** (See page 19-1.)
5. **CLEAR FORCES** (See pages 18-37 and 18-39.)
6. **CLEAR PROG** (See page 19-6.)
7. **COMMS**^① (See page 19-6.)
8. **CONTRAST** (See page 4-18.)

^① If you have configured your program for operation with Series A or B MicroLogix 1000 discrete controllers, this menu option is **BAUD RATE**. (See page 19-7.)

Screen Definition

The following figure shows the menu screen and identifies its main sections.






| Section | Description |
|----------------------|---|
| Menu Options | The list of options available in the menu functional area. These options are described in the manual at their point of use. |
| Selected Menu Option | The option that the flashing arrow is pointing to. |

Menu

How to Complete Tasks

You complete tasks by pressing the appropriate key or key sequence from the menu screen.

| To: | Press: |
|--|---|
| go to a menu option when you know its corresponding number | menu option # |
| choose the selected menu option |  |
| scroll up or down between the menu options |  |
| return to the previous screen |  |

Access Mode by
pressing this key:

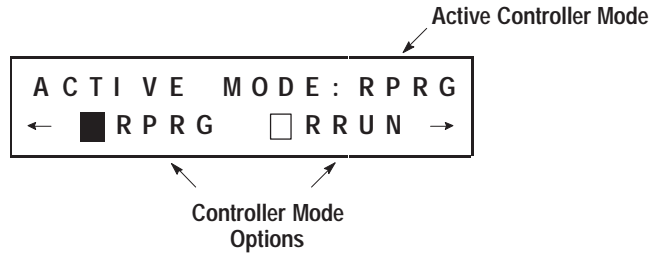


Mode

From the mode functional area, you can change the current mode of the controller.

Screen Definition

The following figure shows the mode screen and identifies its main sections.



| Section | Description |
|-------------------------|---|
| Active Controller Mode | The current mode of the controller is displayed. See the table below for a list of the possible display entries. |
| Controller Mode Options | The controller mode options you can select (RPRG, RRUN, RCSN, and RSSN) are accessed from this screen using the arrow keys. Descriptions of each of these modes and how you change between them can be found beginning on page 18–21. |

The table below shows the possible active controller mode display entries and the corresponding micro controller mode.

| Display Entry | Micro Controller Mode |
|---------------|-------------------------------|
| RPRG | Remote Program |
| RRUN | Remote Run |
| RCSN | Remote Test – Continuous Scan |
| RSSN | Remote Test – Single Scan |
| RSUS | Remote Suspend ^① |
| FLT | Fault ^② |




① The controller only enters suspend mode if you run a program that executes a suspend instruction.

② The controller only enters fault mode if, while a program is executing, a fault occurs within the operating system or the program, or if S1/13 is set at any time. See page 20–11 for information on identifying and clearing faults.

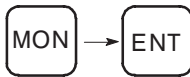
Mode

How to Complete Tasks

You complete tasks by pressing the appropriate key or key sequence from the mode screen.

| To: | Press: |
|--|---|
| scroll left or right between the controller mode options |  |
| choose the controller option that is currently highlighted |  |
| return to the previous screen |  |

Access Program Monitor by pressing these keys:

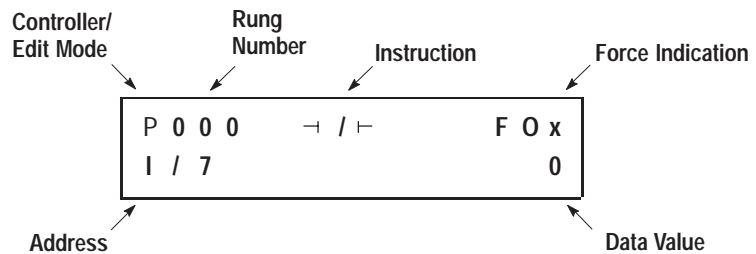


Program Monitor

From the program monitor functional area you can create, view, edit, and troubleshoot your controller programs.

Screen Definition

The following figure shows a typical screen for a bit instruction and identifies its main sections.



| Section | Description |
|----------------------|---|
| Controller/Edit Mode | The current mode of the controller is displayed in abbreviated form, where P=RPRG, R=RRUN, T=RCSN or RSSN, S=RSUS, and F=FLT. If the controller is in RPRG you can utilize the editing modes as well, where P=append and O=overwrite. The mode will flash when edits exist. |
| Rung Number | The rung number currently being viewed is displayed. |
| Instruction | The instruction currently being viewed is displayed. |
| Address | The address currently being viewed. |
| Force Indication | Indicates that the bit currently being viewed is being forced. x=N if forced on, and x=F if forced off. If no force exists, this field is blank. |
| Data Value | The data value of the address is shown here. |

Program Monitor



From Program Monitor you can access these areas:
Menu
Mode
Data Monitor
Multi-Point Function

How to Complete Tasks

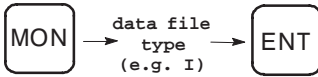
You complete tasks by pressing the appropriate key or key sequence from the program monitor screen.

| To: | Press: |
|--|--------------|
| access the menu options | |
| change the controller's mode (See page 18-23.) | |
| access the multi-point functional area (See page 18-31.) | |
| add a bit address to the next available location in the multi-point list (See page 18-32.) | → |
| execute the trace feature (See page 20-8.) | |
| delete program instructions, or to delete typed characters when entering parameters (See page 17-6.) | |
| delete program rungs, or to delete all typed characters from a line (See page 17-6.) | → |
| add a rung to the current program file after the current rung | |
| force On an external input data file bit or output circuit (See page 18-35.) | |
| force Off an external input data file bit or output circuit (See page 18-35.) | |
| remove a set force from an external input data file bit or output circuit. (See page 18-35.) | → or → |
| search for an instruction or address (See page 17-8.) | |
| toggle the editing mode between overwrite and append (See page 17-3.) | |
| view faults manually (See page 20-11.) | |
| clear a fault manually (See page 20-11.) | → |

Continued on following page

| To: | Press: |
|---|---|
| access the data monitor functional area at the address shown in the screen | MON → ENT |
| move up and down between a program's rungs and program files |  |
| move left and right through each rung of a program. (When the end of a rung is reached, the next rung automatically scrolls into view as you move the cursor right or left in the program.) |  |
| return to the home screen | ESC |
| enter the # character for an indexed address (See page 6–9.) | FUN ^ → ANB 0 |
| access the function code table | FUN ^ → ENT |
| enter data you've typed or confirm a prompt | ENT |

Access Data Monitor by pressing these keys:

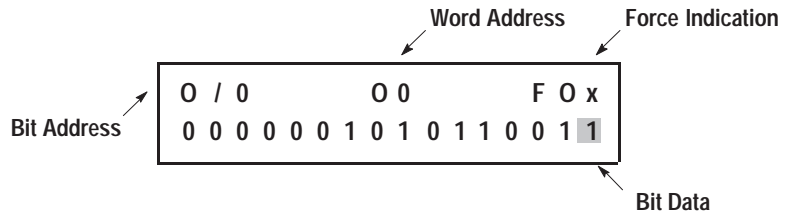


Data Monitor

From the data monitor functional area you can view and edit the data values used in your controller programs. The data is separated into Output, Input, Status, Bit, Timer, Counter, Control, and Integer data files.

Screen Definition

The following figure shows an example of an Output data file and identifies its main sections. (For examples of other data files, see Viewing Data Table Files on page 18–28.)
























| Section | Description |
|------------------|--|
| Bit Address | The address of the bit the cursor is currently on. |
| Word Address | The address of the word currently being viewed. |
| Force Indication | Indicates that the bit currently being viewed is being forced. x=N if forced on, and x=F if forced off. If no force exists, this field is blank. |
| Bit Data | A binary representation of the data. |

**Data
Monitor**

From Data Monitor
you can access
these areas:
Menu
Mode
Program Monitor
Multi-Point Function

How to Complete Tasks

You complete tasks by pressing the appropriate key or key sequence from the data monitor screen.

| To: | Press: |
|--|--|
| access the menu options |  |
| change the controller's mode (See page 18-23.) |  |
| access the multi-point functional area (See page 18-31.) |  |
| add a bit address to the next available location in the multi-point list (See page 18-32.) |  →  |
| execute the trace feature (See page 20-8.) |  |
| delete typed characters on which the cursor is located |  |
| delete all typed characters when entering parameters |  →  |
| force On an external input data file bit or output circuit (See page 18-35.) |  |
| force Off an external input data file bit or output circuit (See page 18-35.) |  |
| remove a set force from an external input data file bit or output circuit. (See page 18-35.) |  →  or  →  |
| search for an instruction or address (See page 17-8.) |  |
| view faults manually (See page 20-11.) |  |
| clear a fault manually (See page 20-11.) |  →  |
| access the program monitor functional area (See page 17-1.) |  →  |

Continued on following page

| To: | Press: |
|--|--------|
| scroll through the data file table | |
| scroll through the bits of individual data files | |
| return to the home screen | ESC |
| change the radix (See page 18–30.) | → |
| enter data you've typed | ENT |

Access Multi-Point Function by pressing this key:

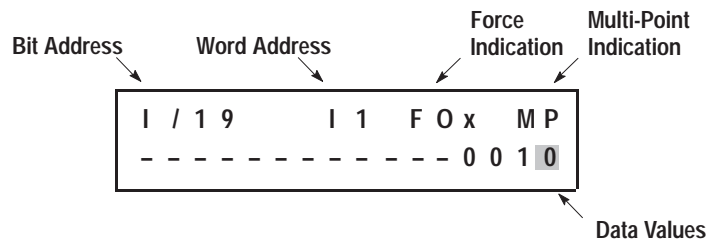


Multi-Point Function

The multi-point function allows you to simultaneously monitor the data of up to 16 non-contiguous bit addresses. Since the multi-point list is stored with the program, you can create a unique list for each program you create. (For more information on using the multi-point function, see page 18–31.)

Screen Definition

The following figure shows a multi-point screen and identifies its main sections.



| Section | Description |
|------------------------|---|
| Bit Address | The address of the bit the cursor is currently on. |
| Word Address | The word address of the bit currently being viewed. |
| Force Indication | Indicates that the bit currently being viewed is being forced. x=N if forced on, and x=F if forced off. If no force exists, this field is blank. |
| Multi-Point Indication | Indicates that you are in the multi-point functional area. |
| Data Values | The data values of the bit addresses assigned to the multi-point list are shown here. Dashed lines indicate that an address has not been assigned to that bit location. |

**Multi-Point
Function**

From Multi-Point Function you can access these areas:
Menu
Mode
Program Monitor
Data Monitor

How to Complete Tasks

You complete tasks by pressing the appropriate key or key sequence from the multi-point screen.

| To: | Press: |
|--|--------|
| access the menu options | |
| change the controller's mode (See page 18-23.) | |
| view faults manually (See page 20-11.) | |
| clear a fault manually (See page 20-11.) | |
| execute the trace feature (See page 20-8.) | |
| delete a single address from the multi-point list (See page 18-34.) | |
| delete all addresses from the multi-point list (See page 18-34.) | |
| force On an external input data file bit or output circuit (See page 18-35.) | |
| force Off an external input data file bit or output circuit (See page 18-35.) | |
| remove a set force from an external input data file bit or output circuit. (See page 18-35.) | |
| search for an instruction or address (See page 17-8.) | |
| access the program monitor functional area (See page 17-1.) | |
| access the data monitor functional area (See page 18-27.) | |
| scroll through the multi-point list | |
| return to the previous screen | |
| enter data you've typed or confirm a prompt | |

Changing the HHPs Defaults

When your MicroLogix 1000 HHP arrives, it has the following factory default settings:

| Feature | Default Setting |
|----------|-----------------|
| Language | English |
| Contrast | ■ ■ ■ ■ □ □ □ □ |

You can use the menu options to change the default settings of these features, as described in the following sections. Any changes you make are saved when power is cycled, so you will not need to set them every time the HHP powers up.

Selecting the Language

You can configure the HHP to display prompts and messages in one of six languages: English, Spanish, German, French, Italian, and Japanese. There are two methods you can use to select a new default setting.

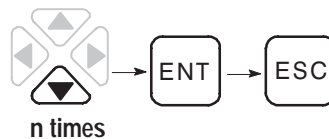
Using the Menu Option

Follow the steps below to change the language using the menu.

1. Access the menu and choose the option 1 . LANGUAGE.









2. Arrow down to the desired language, select it, and return to the previous screen.



Using Short-Cut Keys

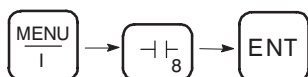
The following table shows the short-cut keys you can press from the home screen to change the language.

| To change the language to: | Press the following keys simultaneously and hold for 1.5 seconds: |
|----------------------------|---|
| English | ESC  |
| Spanish | ESC  |
| German | ESC  |
| French | ESC  |
| Italian | ESC  |
| Japanese | ESC  |

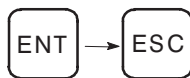
Changing the LCD Display Contrast

Follow the steps below to change the contrast setting for the LCD display.

1. Access the menu and choose the option 8 . CONTRAST.



2. Arrow left or right to select the desired contrast.
3. Enter the selected contrast and return to the previous screen.



Quick Start for New Users

This chapter can help you get started using the MicroLogix 1000 HHP with your micro controller. It provides task-oriented procedures to guide you through a hands-on practice exercise.

Before you begin you should have completed the following tasks:

- Your controller should be installed and wired. (See chapters 1 and 2.)
- Your HHP should be connected and powered-up. (See chapters 3 and 5.)

What to Do First

Here's what you'll be doing to get started with the MicroLogix 1000 HHP:

Preparing to enter a new program

- Placing the controller in program mode
- Clearing the current program

Entering and running the program

- Entering the new program
- Changing to run mode

Monitoring operation

- Monitoring the program
- Monitoring the data

Once you've finished these steps you will have a good idea of what it takes to program your micro controller with a MicroLogix 1000 HHP. You will also know how to execute and monitor program activity.

If You'd Like More Examples

Be sure you read the last section in this chapter. It directs you to more examples throughout the manual.

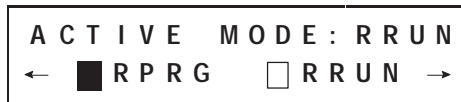
Preparing to Enter a New Program

Before you can enter a new program, you must complete the two preliminary procedures described in this section.

Placing the Controller in Program Mode

If the controller is not currently in program mode, you need to change to that mode. Follow the steps below:

1. From the home screen, access the mode options.



2. Select RPRG mode. (The RPRG mode box is already highlighted.)



Once the controller enters the program mode, you are returned to the home screen.

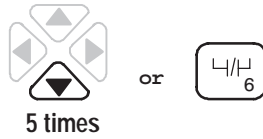
Clearing the Current Program

Clear the current program from the controller by following these steps:

1. Access the menu screen.



2. Arrow down to menu option 6, or press the number 6.



```
→ 6 . CLEAR PROG
   7 . BAUD RATE
```

3. Select menu option 6.



```
CLEAR PROGRAM?
YES [ENT] NO [ESC]
```

4. Clear the program in the controller.



```
CLEAR PROGRAM?
CLEARING . . . . .
```

5. Return to the home screen.



Once the program is cleared, the home screen shows the default program name MICRO.

Reviewing What You've Done So Far

You have completed preparing to enter a new program with your HHP.

- Preparing to enter a new program**
 - Placing the controller in program mode
 - Clearing the current program
- Entering and running the program**
 - Entering the new program
 - Changing to run mode
- Monitoring operation**
 - Monitoring the program
 - Monitoring the data

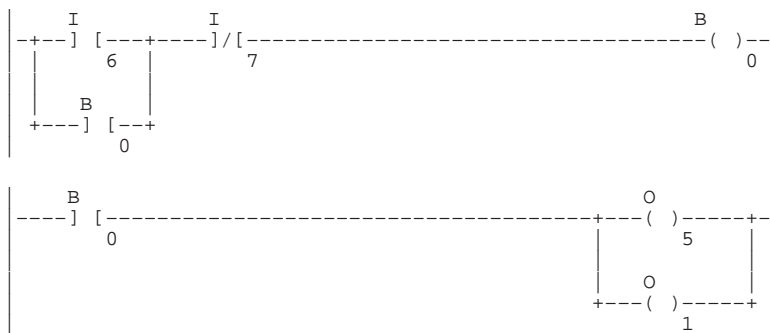
Continue on to the next section to enter and run a program.

Entering and Running the Program

You are now ready to create a program in file 2. Once the program is entered, you will place the controller in run mode so you can monitor the program.

Entering the New Program

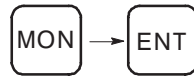
The following rungs consist of LD, OR, ANI, and OUT instructions. You will learn about these instructions in chapters 6 and 13. For now though, we will use them to give you an idea of how to enter a simple program using the MicroLogix 1000 HHP. The following diagram is the ladder representation of what you will enter in the HHP.



Enter the rungs by completing the steps that follow.

Important: If you make an error at any time, you can abandon the operation by pressing the **ESC** key.

1. From the home screen, access the program monitor display for the program MICRO.



```
P  S T A R T   F I L E : 0 2
M A I N _ P R O G
```

The start of file screen appears. This is where you start inserting the program rungs.

2. Insert a rung in file 2, the main program file.

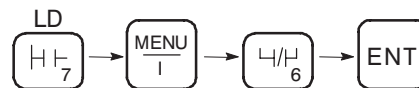
Remember, the MicroLogix 1000 HHP is designed to distinguish which function you want to perform based on the context you are in at the time you press a key. Therefore, pressing the key shown below will automatically access a new rung, and not a T.



```
P 0 0 0  |
```

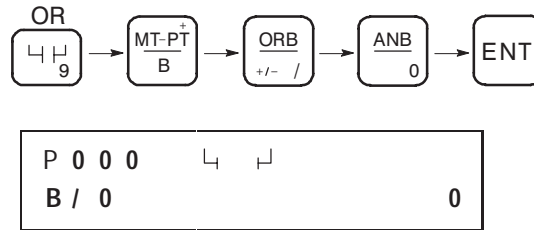
The P in the upper left-hand corner is flashing because you are making changes to the program.

3. Enter the first normally open instruction (LD) on the rung. For the input file type, the / character is automatically displayed by the HHP.

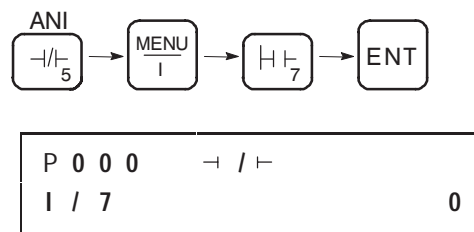


```
P 0 0 0  | I  |
I / 6                                0
```

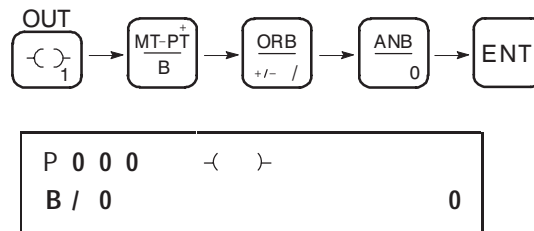
4. Place a normally open instruction in parallel (OR) with the first one.



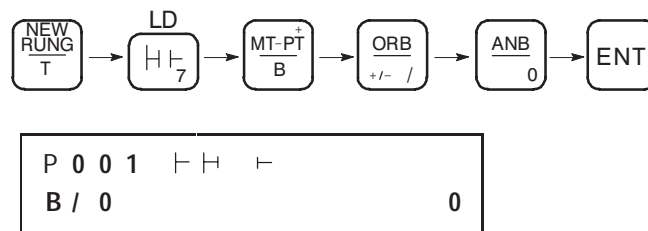
5. Enter a normally closed instruction in series (ANI) with the first two.



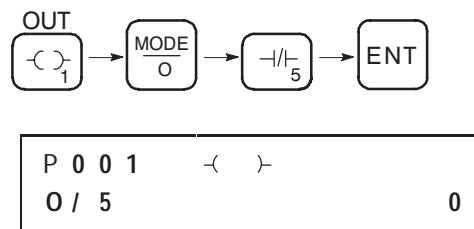
6. Enter the first output instruction (OUT) on the rung.



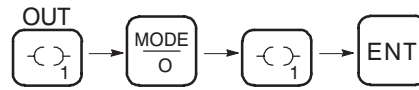
7. Start a new rung after the first one, and enter another LD instruction.



8. Add an output instruction to the rung. As with entering the input file type, the HHP automatically shows the / character for the output file type.



9. Add the final output instruction to the rung.



```
P 0 0 1    ( )
O / 1                                0
```

10. Return to the home screen.



```
M I C R O                R P R G
F R E E : * * *    F I L E : 0 2
```

The RPRG is flashing because edits exist. Also, the number of free instruction words is not known until the program is checked, so three asterisks are displayed.

Changing to Run Mode

Now that you have entered a program, you can run it by changing to run mode. Verify the mode by looking in the upper right-hand corner of the HHP display. Right now it reads RPRG (remote program mode). To change into remote run mode, RRUN, follow these steps:

1. Access the mode options.



```
A C T I V E   M O D E : R P R G
←  ■ R P R G   □ R R U N  →
```

2. Arrow right to RRUN.



```
ACTIVE MODE: RPRG
←  □ RPRG  ■ RRUN  →
```

3. Select remote run mode. The program is checked and, if accepted, the home screen appears. If you get a fault code, refer to chapter 20 to clear the fault.



```
M I C R O           R R U N
F R E E : 7 2 9   F I L E : 0 2
```

RRUN now appears in the upper right-hand corner of the screen. Also, the number of free instruction words is displayed.

Reviewing What You've Done So Far

You are now monitoring the program file.

- Preparing to enter a new program**
 - Placing the controller in program mode
 - Clearing the current program
- Entering and running the program**
 - Entering the new program
 - Changing to run mode
- Monitoring operation**
 - Monitoring the program
 - Monitoring the data

Continue on with the next section to monitor the operation of your program.

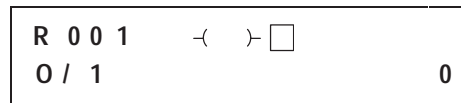
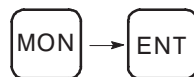
Monitoring Operation

You can monitor the operation of your program by viewing the program files and the data files.

Monitoring the Program

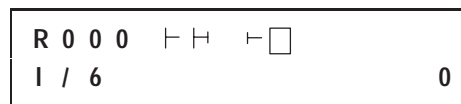
You should now be running the program MICRO. You can test the operation of your program by monitoring the relay instruction states. Instruction state boxes appear to the right of each bit instruction. When filled, these boxes indicate that logical continuity exists in the program.

1. From the home screen, access the program monitor.



You return to the last location you were at within the program. The instructions will either be on or off, depending on the input and output states in your program.

2. Arrow back to bit I/6 on rung 0 of the program. As you do this, look at the instruction state boxes and see which ones, if any, are filled.



Currently, I/6 is off. If I/6 is turned on, the instruction state box will be filled. This will result in a path of logical continuity in the rung, causing the output instruction state box to be filled as well.

Monitoring the Data

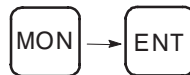
Next you will monitor the input and output data files. These files contain bits corresponding to the I/O screw terminals of the controller.

1. From the program monitor, go to rung 1. You can access this rung by entering the rung number as shown here:



```
R 0 0 1  | H | H | □
B / 0                                     0
```

2. Access the data monitor for the first instruction on the rung (B/0).



```
B / 0      B 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

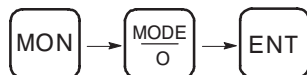
3. Set the cursored bit to 1.



```
B / 0      B 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

Notice that the bit is set to 1 as soon as the key is pressed.

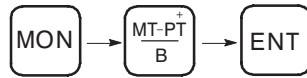
4. Now access the output data file.



```
O / 0      O 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0
```

Notice that bits O/1 and O/5 are set to 1. These bits turned on when you set bit B/0 to 1.

5. Return to the data word B/0.



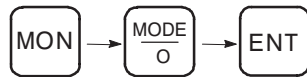
| | |
|-----------------------------|-----|
| B / 0 | B 0 |
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 1 |

6. Reset the cursored bit to 0.



| | |
|-----------------------------|-----|
| B / 0 | B 0 |
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 |

7. Return to the output data file.



| | |
|-----------------------------|-----|
| O / 0 | O 0 |
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 |

Notice that bits O/1 and O/5 are set to 0. These bits turned off when you reset bit B/0 to 0.

8. Return to the home screen.



| | |
|-----------------|---------------|
| M I C R O | R R U N |
| F R E E : 7 2 8 | F I L E : 0 2 |

Reviewing What You've Done So Far

Congratulations! You have finished entering, running, and monitoring a sample program using the MicroLogix 1000 HHP.

Preparing to enter a new program

- Placing the controller in program mode
- Clearing the current program

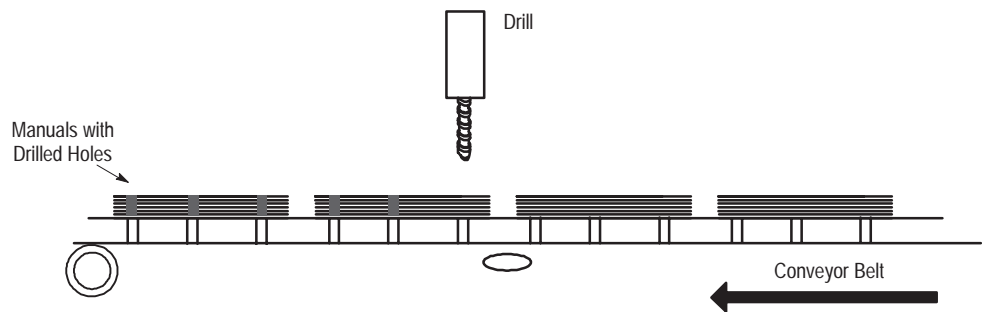
Entering and running the program

- Entering the new program
- Changing to run mode

Monitoring operation

- Monitoring the program
- Monitoring the data

The program MICRO that you created in this chapter is actually part of a bigger application example provided in appendix D called Paper Drilling Machine. In that application the rungs you entered for MICRO control the movement of the conveyor belt and drill bit shown below.



What to Do Next

If you want more hands-on experience, chapters 8 through 14 show you more portions of this application example. If you follow through and add the rungs provided at the end of each of those chapters, you will have the complete program entered by the time you reach the end of chapter 14. Appendix D contains the complete example and its description.

Also, overviews are provided at the beginning of chapters 8 through 14 to introduce you to the concepts you'll learn in each chapter.

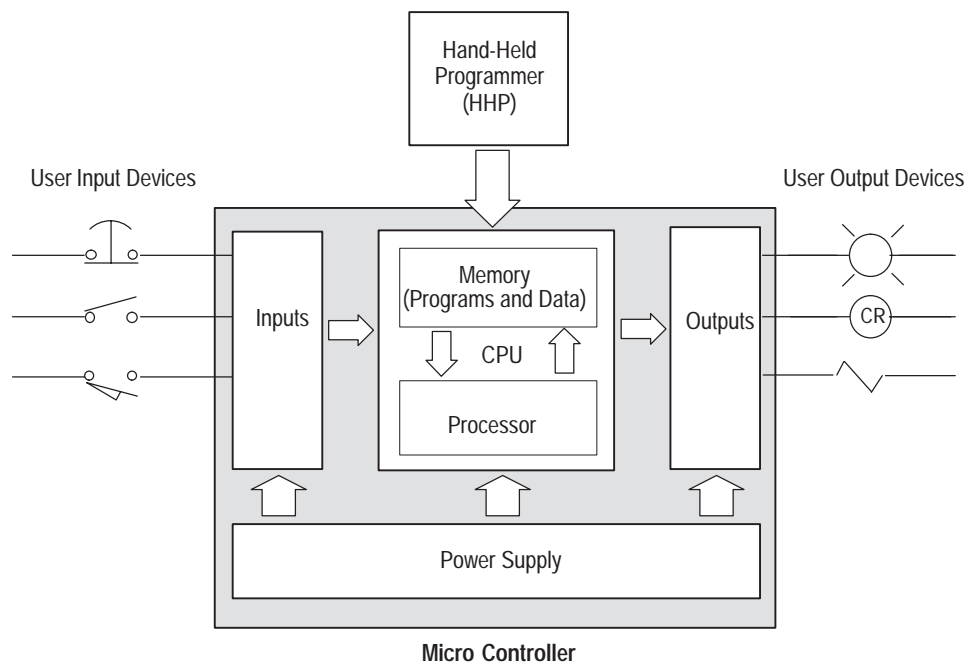
Programming Overview

This chapter explains how to use the MicroLogix 1000 HHP to program the micro controller. Read this chapter for basic information about:

- principles of machine control
- understanding file organization
- understanding how programs are stored and accessed
- addressing data files
- applying logic to your schematics
- a model for developing your program

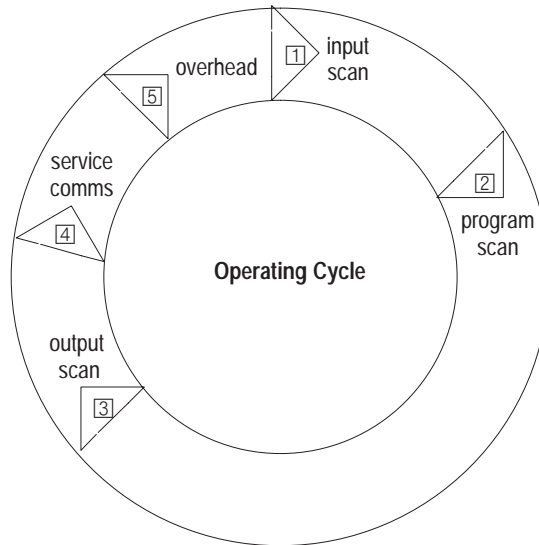
Principles of Machine Control

The controller consists of a built-in power supply, central processing unit (CPU), inputs, which you wire to input devices (such as push buttons, proximity sensors, limit switches), and outputs, which you wire to output devices (such as motor starters, solid-state relays, and indicator lights).



You enter a logic program into the controller using the HHP. The logic program is based on your electrical relay print diagrams. It contains instructions that direct control of your application.

With the logic program entered into the controller, placing the controller in the Run mode initiates an operating cycle. The controller's operating cycle consists of a series of operations performed sequentially and repeatedly, unless altered by your program logic.



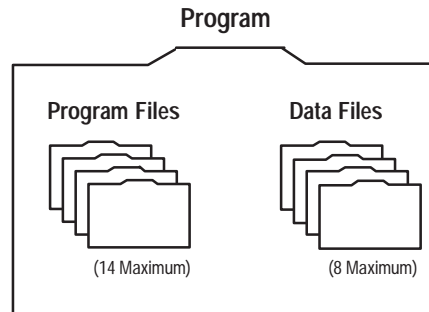
- ① input scan – the time required for the controller to scan and read all input data; typically accomplished within μ seconds.
- ② program scan – the time required for the controller to execute the instructions in the program. The program scan time varies depending on the instructions used and each instruction's status during the scan time.

Important: Subroutine and interrupt instructions within your logic program may cause deviations in the way the operating cycle is sequenced.

- ③ output scan – the time required for the controller to scan and write all output data; typically accomplished within μ seconds.
- ④ service communications – the part of the operating cycle in which communication takes place with other devices, such as a MicroLogix 1000 HHP or a personal computer.
- ⑤ housekeeping and overhead – time spent on memory management and updating timers and internal registers.

Understanding File Organization

The micro controller provides control through the use of a program. Most of the operations you perform with the MicroLogix 1000 HHP involve the program and the two components created with it: program files and data files.

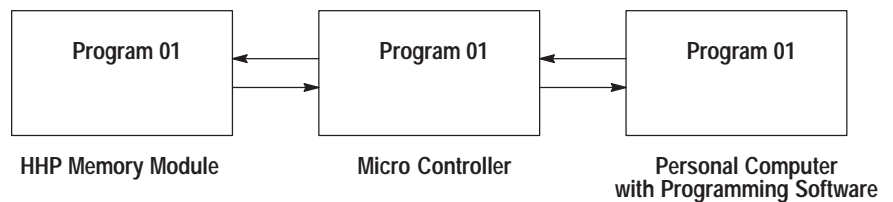


Notes on terminology: The term *program* used in HHP displays is equivalent to the term *processor file* that may be used in some programming software packages.

Program

A program consists of the collective program files and data files. It contains all the instructions, data, and configuration information pertaining to that program.

The program is located in the micro controller. It can be transferred to/from a memory module (optional) located in the HHP, or to/from a personal computer with programming software.



Program Files

Program files contain controller information, the main program, interrupt subroutines, and any subroutine programs. These files are:

- **System Program** (file 0) – This file contains various system related information and user-programmed information such as program name and password.
- **Reserved** (file 1) – This file is reserved.
- **Main Program** (file 2) – This file contains user-programmed instructions defining how the controller is to operate.
- **User Error Fault Routine** (file 3) – This file is executed when a fault occurs. It is only used for this purpose.
- **High-Speed Counter Interrupt** (file 4) – This file is executed when an HSC interrupt occurs. It can also be used for a subroutine program.
- **Selectable Timed Interrupt** (file 5) – This file is executed when an STI occurs. It can also be used for a subroutine program.
- **Subroutine Program** (files 6 – 15) – These are used according to subroutine instructions residing in the main program file or other subroutine files.

Data Files

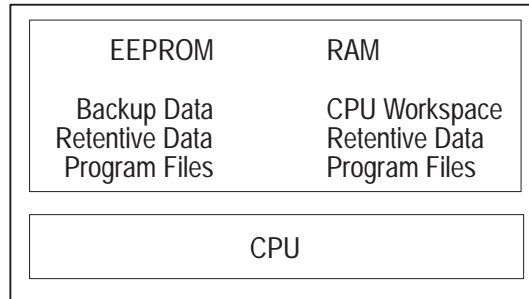
Data files contain the status information associated with external I/O and all other instructions you use in your main and subroutine program files. In addition, these files store information concerning controller operation. You can also use the files to store “recipes” and look-up tables if needed.

These files are organized by the type of data they contain. The data file types are:

- **Output** – This file stores the state of the output terminals for the controller.
- **Input** – This file stores the status of the input terminals for the controller.
- **Status** – This file stores controller operation information. It is useful for troubleshooting controller and program operation.
- **Bit** – This file is used for internal relay logic storage.
- **Timer** – This file stores the timer accumulator and preset values and the status bits.
- **Counter** – This file stores the counter accumulator and preset values and the status bits.
- **Control** – This file stores the length, position pointer, and status bits for specific instructions such as bit shifts and sequencers.
- **Integer** – This file is used to store numeric values or bit information.

Understanding How Programs are Stored and Accessed

The micro controller uses two devices for storing programs: RAM and EEPROM. The RAM provides short-term storage (i.e., its data is lost on a power down), while the EEPROM provides long-term storage (i.e., its data is not lost on a power down). The diagram below shows how the memory is allocated in the micro controller's processor.



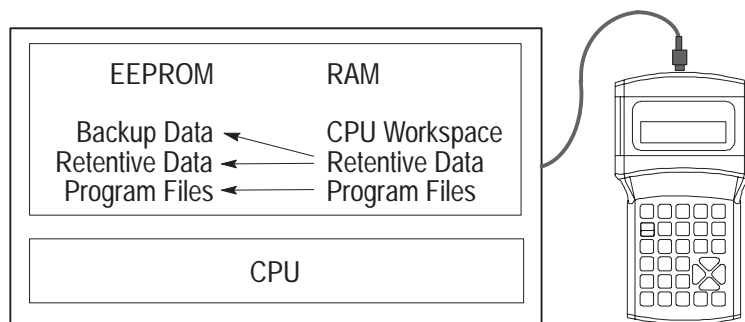
The memory device that is used depends on the operation being performed. This section describes how memory is stored and accessed during the following operations:

- saving a program
- normal operation
- power down
- power up

Saving a Program

A program is saved whenever you accept edits or change from RPRG mode to RRUN, RCSN, or RSSN mode when program edits exist. When the program is saved, it is first stored in the volatile RAM. It is then transferred to the non-volatile EEPROM, where it is stored as both backup data and retentive data.

Important: If no edits exist and you select the `ACCEPT EDITS` menu option, no backup data is written to the controller. It remains unchanged.

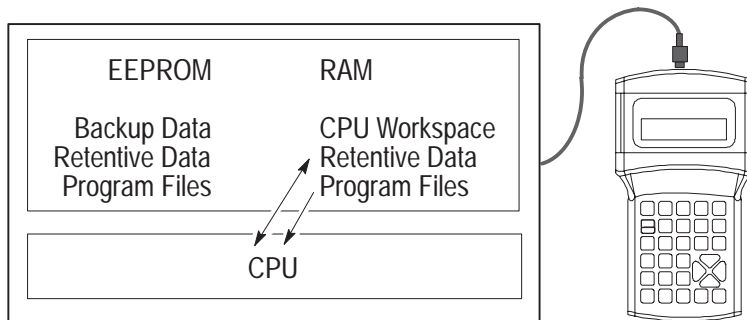


Important: If you want to ensure that the backup data is the same for every micro controller you are using, store the program in the memory module before saving it. Then, when making edits to the program, load the program from the memory module.

Normal Operation

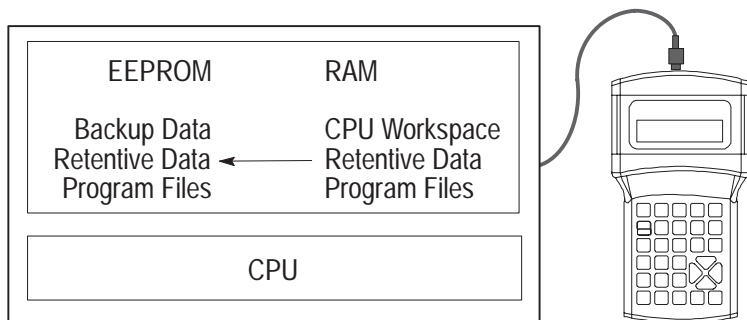
During normal operation, both the micro controller and the HHP access the program stored in the RAM. Any changes to retentive data that occur due to program execution or programming commands affect only the retentive data in the RAM.

The program files are never modified during normal operation. However, both the CPU and the HHP can read the program files stored in RAM.



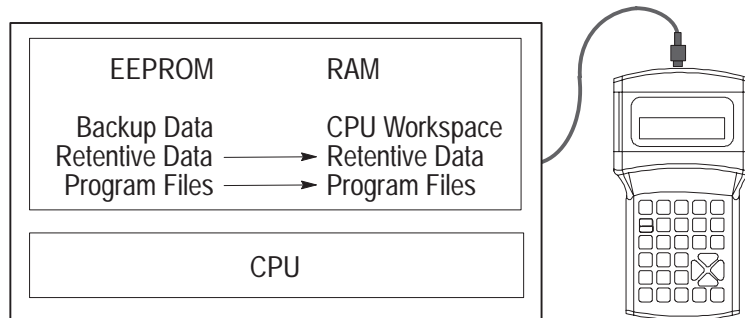
Power Down

When a power down occurs, only the retentive data is transferred from the RAM to the EEPROM. (The program files do not need to be saved to the EEPROM since they cannot be modified during normal operation.) If for some reason power is lost before all of the retentive data is saved to the EEPROM, the retentive data is lost. This may occur due to an unexpected reset or a hardware problem.

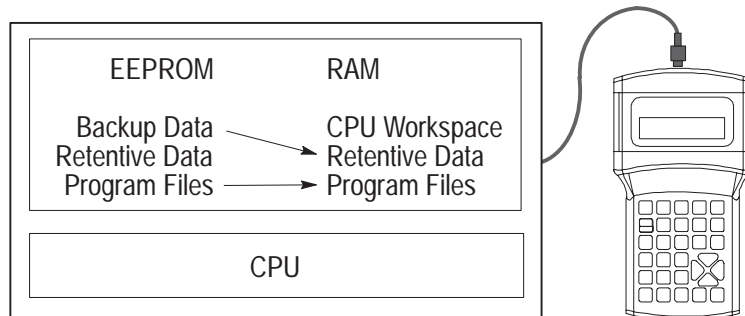


Power Up

During power up, the micro controller transfers the program files from the EEPROM to the RAM. The retentive data is also transferred to the RAM, provided it was not lost on power down, and normal operation begins.



If retentive data was lost on power down, the backup data from the EEPROM is transferred to the RAM and used as the retentive data. In addition, status file bit S5/8 (retentive data lost) is set and a recoverable major error occurs when the mode is changed to RRUN, RCSN, or RSSN.



Addressing Data Files

For the purposes of addressing, each data file type is identified by a letter (identifier) and has an associated allowable range for data storage.

| File Type | Identifier | Allowable Word Range, 0 to: | |
|-----------|------------|-----------------------------|--------|
| | | Discrete | Analog |
| Output | O | 0 | 4 |
| Input | I | 1 | 7 |
| Status | S | 32 | – |
| Bit | B | 31 | – |
| Timer | T | 39 | – |
| Counter | C | 31 | – |
| Control | R | 15 | – |
| Integer | N | 104 | – |

The addresses are made up of alphanumeric characters separated by delimiters. Delimiters include the slash and period.

Specifying Logical Addresses

You assign logical addresses to instructions from the highest level (element) to the lowest level (bit). Addressing examples are shown in the table below.

| To specify the address of a: | Use these parameters: ^① |
|------------------------------|--|
| Word within an integer file | |
| Word within a structure file | |
| Bit within an integer file | |
| Bit within a bit file | <p>Bit files are bit stream continuous files; therefore, you can address them in two ways: by bit alone (B/31), or by word and bit (B1/5).</p> |
| Bit within a structure file | |

^① The addressing parameters required when using the MicroLogix 1000 HHP differ slightly from those required when using programming software.

You must address at the bit level using mnemonics for timer, counter, or control data types. The available mnemonics depend on the type of data. See chapters 8 through 14 for more information.

Specifying Indexed Addresses

The indexed address symbol is the # character, which is placed immediately before the file-type identifier in a logical address. You can use more than one indexed address in your ladder program.

When you specify indexed addresses, follow these guidelines:

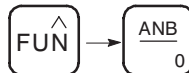
- Make sure the index value (positive or negative) does not cause the indexed address to exceed the file type boundary.
- When an instruction uses more than two indexed addresses, the controller uses the same index value for each indexed address.
- Set the index word to the offset value you want immediately before enabling an instruction that uses an indexed address. The controller starts operation at the base address plus the offset.



ATTENTION: Instructions with a # sign in an address manipulate the offset value stored at S24. Make sure you monitor or load the offset value you want prior to using an indexed address. Otherwise unpredictable machine operation could occur with possible damage to equipment and/or injury to personnel.

Entering the # Character

When entering addresses, you can access the indexed address symbol by pressing the key sequence shown below:



Example of Indexed Addressing

As an example, a Masked Move (MVM) instruction uses an indexed address in the source and destination addresses. If the offset value is 10 (stored in S24), the controller manipulates the data stored at the base address plus the offset. See the table below.

| Value: | Base Address: | Offset Value in S24: | Offset Address: |
|-------------|---------------|----------------------|-----------------|
| Source | N10 | 10 | N20 |
| Destination | N50 | 10 | N60 |

Addressing File Instructions – Using the File Indicator (#)

The file instructions shown below manipulate data table files. These instructions are addressed with the # sign. They store an offset value in word S24 (index register), just as with indexed addressing discussed in the last section.

| | | | |
|------------|-----------------|------------|-------------------|
| COP | Copy File | LFL | (LIFO Load) |
| FLL | Fill File | LFU | (LIFO Unload) |
| BSL | Bit Shift Left | SQO | Sequencer Output |
| BSR | Bit Shift Right | SQC | Sequencer Compare |
| FFL | (FIFO Load) | SQL | Sequencer Load |
| FFU | (FIFO Unload) | | |

When entering any of the instructions shown above, the # character is automatically inserted for you.



ATTENTION: If you are using file instructions and also indexed addressing, make sure that you monitor and/or load the correct offset value prior to using an indexed address. Otherwise, unpredictable operation could occur, resulting in possible personal injury and/or damage to equipment.

Entering Numeric Constants

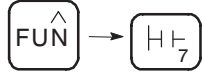
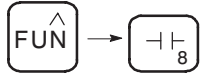
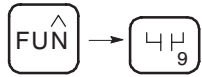
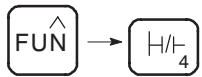
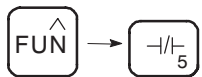
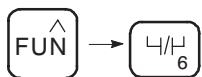
You can enter numeric constants directly into many of the instructions you program. They are used in place of data file elements.

The range of values for most instructions is –32,768 through +32,767. These values may be entered and displayed in one of several radixes:

- Decimal
- Hexadecimal
- Binary

The radix default that is used is determined by the operand (e.g., presets are always displayed as decimal values and masks are always displayed as hexadecimal values).

When entering hexadecimal characters, you may need to access the following additional characters, not displayed on the keypad.

| To Access This Character: | Press This Key Sequence: |
|---------------------------|---|
| A |  |
| B |  |
| C |  |
| D |  |
| E |  |
| F |  |

You can change the radix for output, input, bit and integer data files. See page 18–30 for more information.

Applying Logic to Your Schematics

The program is made from the logic you enter into the micro controller. Ladder logic instruction programming, which you may already be familiar with, is one method used to enter logic. Another method is called Instruction List (Boolean) programming. Your MicroLogix 1000 HHP uses this type of logic programming. An overview of ladder logic and instruction list programming is provided below.

Understanding Ladder Logic Instruction Programs

Ladder logic is a graphical programming language based on electrical relay diagrams. Instead of having electrical rung continuity, ladder logic is looking for logical rung continuity. A ladder diagram identifies each of the elements in an electromechanical circuit and represents them graphically. This allows you to see how your control circuit operates before you actually start the physical operation of your system.



In a ladder diagram, each of the input devices are represented in series or parallel combinations across the rung of the ladder. The last element on the rung is the output that receives the action as a result of the conditional state of the inputs on the rung.

Each output instruction is executed by the controller when the rung is scanned and the conditions on the rung are true. When the rung is not scanned or the logic conditions on the rung do not create a true logic path, the output is not executed. The maximum number of instructions per rung is 128.

Series Connections

One form of logical continuity is series (AND) logic. This means that when all input conditions in the path are true, energize the output.



In the above example, if A *and* B are true, energize C.

Parallel Connections

Another form of logical continuity is parallel (OR) logic. This means that when one or another path of logic is true, energize the output.

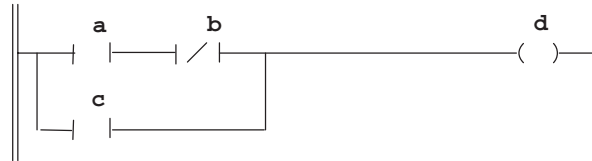


In the above example, if A *or* B is true, energize C.

Parallel logic is formed using branches in your program. Branches can be established at both input and output portions of a rung. The upper limit on the number of levels which can be programmed in a branch structure is 75.

Input Branching

Use an input branch in your application program to allow more than one combination of input conditions to form parallel branches (OR-logic conditions). If at least one of these parallel branches forms a true logic path, the rung logic is enabled. If none of the parallel branches forms a true logic path, rung logic is not enabled and the output instruction logic will not be true. The output is not energized.



In the above example, either *A and B*, or *C* provides a true logical path.

Output Branching

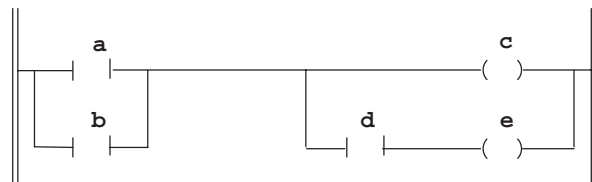
You can program parallel outputs on a rung to allow a true logic path to control multiple outputs. When there is a true logic path, all parallel outputs become true.



In the above example, either *A or B* provides a true logical path to all three output instructions.

Additional input logic instructions (conditions) can be programmed in the output branches to further control the outputs. When there is a true logic path, including extra input conditions on an output branch, that branch becomes true. For instruction list programming, MPS, MRD, and MPP instructions are sometimes needed for this connection. (See page 8–10.)

Example — Parallel Output Branching with Conditions

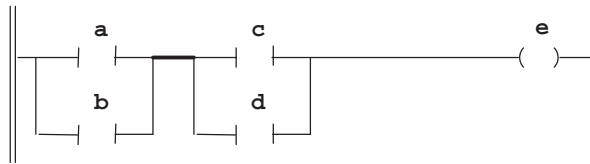


In the above example, either *A and D or B and D* provide a true logic path to *E*.

Connecting Blocks

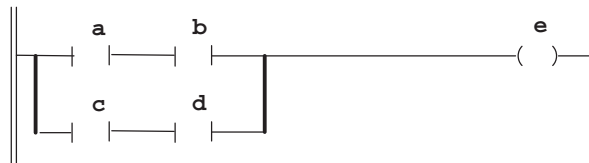
Blocks of input and output instructions can be connected in series and parallel as well.

Example – Series Block Connection



In the above example, two blocks of information are connected in series. Either *A or B*, and *C or D* provides a true logical path. For instruction list programming, the ANB instruction represents this connection. (See page 8–12.)

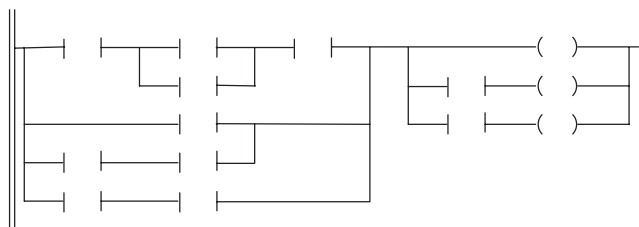
Example – Parallel Block Connection



In the above example, two blocks of information are connected in parallel. Either *A and B*, or *C and D* provides a true logical path. For instruction list programming, the ORB instruction represents this connection. (See page 8–12.)

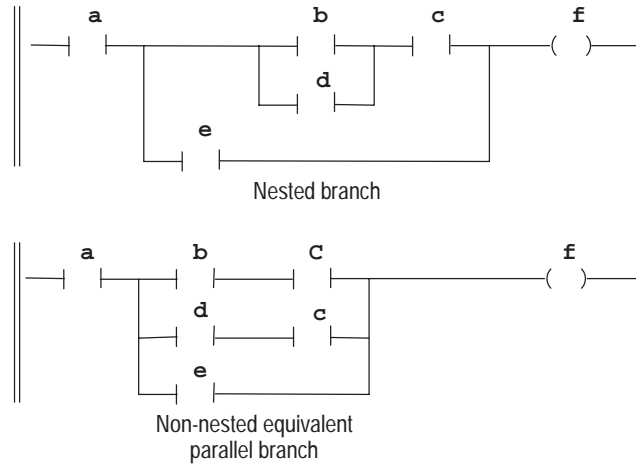
Nested Branching

Input and output branches can be “nested” to avoid redundant instructions, to speed-up controller evaluation, and to provide more efficient programming. A “nested” branch is a branch that starts or ends within another branch. You can nest branches up to four levels deep.



Nested branching can be converted into non-nested branches by repeating instructions to make parallel equivalents.

Example – Non-Nested Equivalent



Understanding Instruction List Programs

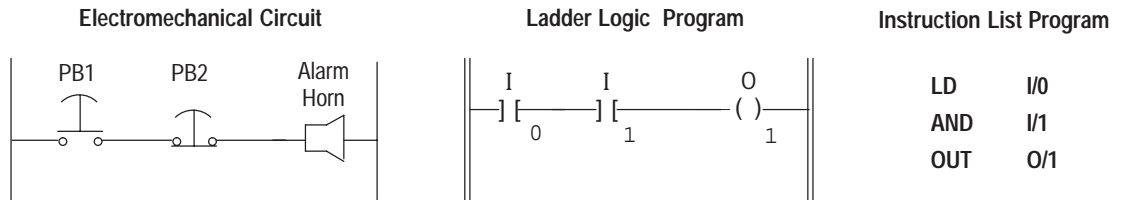
Instruction list (Boolean) programming uses *mnemonics* to represent all the functions and connections available in a ladder logic instruction set. Mnemonics are instructions written in abbreviated form, using two or three letters that imply the operation of the instruction, such as LD, AND, and OUT.

| | | |
|-----|-----|----------------------|
| LD | I/0 | } input instructions |
| AND | I/1 | |
| OUT | O/1 | output instruction |

The MicroLogix 1000 HHP allows you to enter an instruction list program when programming the micro controller.

Applying Ladder Logic and Instruction List

In the following illustration, the electromechanical circuit shows PB1 and PB2, two push buttons, wired in series with an alarm horn. PB1 is a normally open push button and PB2 is normally closed. This same circuit is shown in ladder logic by two contacts wired in series with an output. Contact I/0 and I/1 are examine-if-closed instructions.^① (For more information on this instruction, refer to page 8–3.)



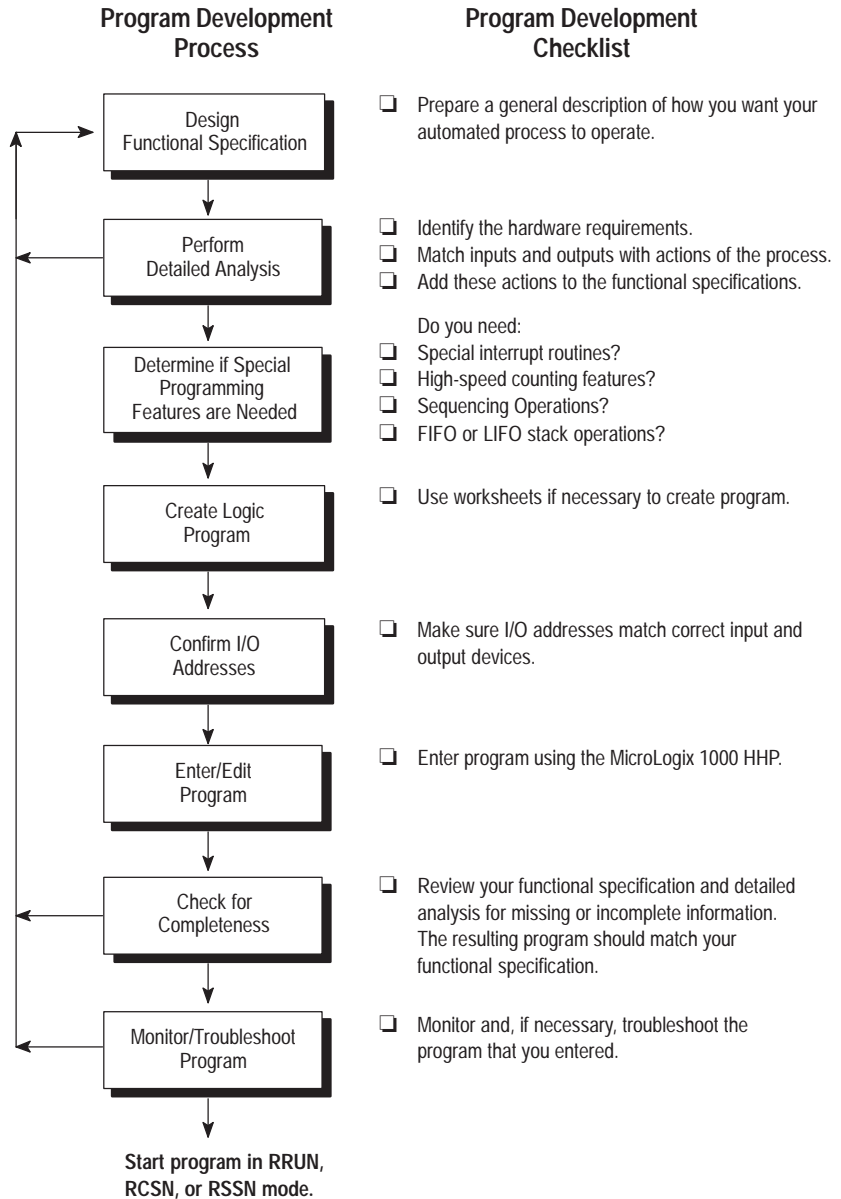
^① Contact I1 would be an examine-if-open instruction (/I) if PB2 was a normally open electromechanical circuit.

The table below shows how these circuits operate. The table shows all possible conditions for the electromechanical circuit, the equivalent state of the instructions, and the resulting output state.

| If PB1 is: | I/0 state is: | And PB2 is: | I/1 state is: | Then the Alarm Horn (O/1) is: |
|------------|---------------|-------------|---------------|-------------------------------|
| not pushed | 0 | not pushed | 1 | silent |
| not pushed | 0 | pushed | 0 | silent |
| pushed | 1 | not pushed | 1 | alarm |
| pushed | 1 | pushed | 0 | silent |

Developing Your Logic Program – A Model

The following diagram can help you develop your application program. Each process block represents one phase of program development. Use the checklist at the right of the process block to help you identify the tasks involved with each process.



Using Analog

This chapter describes the operation of the MicroLogix 1000 analog controllers. Topics include:

- I/O Image
- I/O Configuration
- Input Filter and Update Times
- Converting Analog Data

I/O Image

The input and output image files of the MicroLogix 1000 analog controllers have the following format:

| Address | Input Image | Output Image | Address |
|---------|--------------------------|--------------------------------------|---------|
| I:0.0 | Discrete Input Word 0 | Discrete Output Word 0 | O:0.0 |
| I:0.1 | Discrete Input Word 1 | Reserved | O:0.1 |
| I:0.2 | Reserved | Reserved | O:0.2 |
| I:0.3 | Reserved | Reserved | O:0.3 |
| I:0.4 | Analog Input 0 (Voltage) | Analog Output 0 (Voltage or Current) | O:0.4 |
| I:0.5 | Analog Input 1 (Voltage) | | |
| I:0.6 | Analog Input 2 (Current) | | |
| I:0.7 | Analog Input 3 (Current) | | |

Input words 0 and 1 contain discrete input data. Unused inputs in the discrete inputs image space are reset during each input scan. Input words 2 and 3 are reserved and are not updated by the controller. These inputs have no direct effect on controller operation, but can be modified like other data bits.

Input words 4–7 contain the status of the four analog input channels respectively. Analog input image words are cleared at Going To Run (GTR). For enabled channels, the analog input image is updated on a cyclical basis.

Output word 0 contains discrete output data. Output words 1–3 are reserved output image space. Unused outputs in both the discrete output image space and the reserved output image space have no direct effect on controller operation. But these outputs can be modified like other data bits. Output word 4 holds the value of the analog output channel.

I/O Configuration

The analog input channels are single-ended (unipolar) circuits and can be individually enabled or disabled. The default is all input channels enabled. The two voltage inputs accept $\pm 10.5\text{V}$ dc and the two current inputs accept ± 21 mA.

The analog output channel is also a single-ended circuit. You can configure either voltage (0V dc to +10V dc) or current (+4 to +20 mA) output operation. The default is voltage output.

The output must be configured for either voltage *or* current, *not both*. This is determined by the output configuration. When in the Run mode and the output is configured for voltage, the voltage output terminal is active and the current output terminal is inactive. Similarly, when in the Run mode and the output is configured for current, the current output terminal is active and the voltage output terminal is inactive. When the system is not in Run mode, both the voltage and current outputs are inactive. Refer to page 18–14 for more information on how to configure the analog inputs and outputs with the MicroLogix 1000 HHP.

Input Filter and Update Times

The MicroLogix analog input filter is programmable. The slower the filter setting, the more immune the analog inputs are to electrical noise. The more immune the analog inputs are to electrical noise, the slower the inputs are to update. Similarly, the faster the filter setting, the less immune the analog inputs are to electrical noise. The less immune the analog inputs are to electrical noise, the faster the inputs are to update.

The selected setting is applied to all four analog input channels. The settings supported are shown in the following table. Both the analog input resolution and the analog input settling time are a function of the input filter selection.

| Programmable Filter Characteristics | | | | |
|-------------------------------------|----------------------------------|--------------------|----------------------|-------------------|
| 1st Notch Freq (Hz) | Filter Bandwidth (-3 dB Freq Hz) | Update Time (mSec) | Settling Time (mSec) | Resolution (Bits) |
| 10 | 2.62 | 100.00 | 400.00 | 16 |
| 50 | 13.10 | 20.00 | 80.00 | 16 |
| 60 ^① | 15.72 | 16.67 | 66.67 | 16 |
| 250 | 65.50 | 4.00 | 16.00 | 15 |

^① 60 Hz is the default setting.

The total update time for each channel is a combination of the Update Time and the Settling Time. When more than one analog input channel is enabled, the maximum update for each channel is equal to one ladder scan time plus the channel's Update Time plus Settling Time. When only one analog input channel is enabled, the maximum update for the channel is equal to the Update Time plus one ladder scan time.

Update Time Examples

Example 1 – All (4) channels enabled with 60 Hz filter selected (default settings).

$$\begin{aligned}\text{Maximum Update Time} &= (4) \times \text{ladder scan time} \\ &+ (4) \times 16.67 \text{ ms} \\ &+ (4) \times 66.67 \text{ ms} \\ &= 333.36 \text{ ms} + (4) \times \text{ladder scan times}\end{aligned}$$

(Each channel is updated approximately three times per second.)

Example 2 – 1 channel enabled with 250 Hz filter selected.

$$\text{Maximum Update Time} = \text{ladder scan time} + 4 \text{ ms}$$

Input Channel Filtering

The analog input channels incorporate on-board signal conditioning. The purpose of this conditioning is to reject the AC power line noise that can couple into an analog input signal while passing the normal variations of the input signal.

Frequency components of the input signal at the filter frequency are rejected. Frequency components below the filter bandwidth (–3 dB frequency) are passed with under 3 dB of attenuation. This pass band allows the normal variation of sensor inputs such as temperature, pressure and flow transducers to be input data to the processor.

Noise signals coupled in at frequencies above the pass band are sharply rejected. An area of particular concern is the 50/60 Hz region, where pick-up from power lines can occur.

Converting Analog Data

The analog input circuits are able to monitor current and voltage signals and convert them to digital data. There are six terminals assigned to the input channels that provide two voltage inputs, two current inputs, and two return signals (commons).

The analog outputs can support either a current *or* voltage function. There are three terminals assigned to the output channels that provide one voltage output, one current output, and a common (shared) terminal.

The following table shows sample Analog Signal and Data Word values using the nominal transfer function formula:

$$N = I_{in} \times 32767/21 \quad \text{where } I_{in} \text{ (analog signal) is in milliamperes (mA)}$$

$$N = V_{in} \times 32767/10.5 \quad \text{where } V_{in} \text{ (analog signal) is in volts (V)}$$

$$N = (I_{out} - 4 \text{ mA}) \times 32767/16 \text{ mA} \quad \text{where } I_{out} \text{ (analog signal) is in milliamperes (mA)}$$

$$N = V_{out} \times 32767/10V \quad \text{where } V_{out} \text{ (analog signal) is in volts (V)}$$

| Analog Signal | Data Word | |
|---------------|-----------|--------|
| | Input | Output |
| 0V | 0 | 0 |
| 5V | 15603 | 16384 |
| 10V | 32107 | 32767 |
| 4 mA | 6241 | 0 |
| 11 mA | 17164 | 14336 |
| 20 mA | 31207 | 32767 |

Converting Analog Input Data

Analog inputs convert current and voltage signals into 16-bit two's complement binary values.

To determine an approximate voltage that an input value represents, use one of the following equations:

$$\frac{10.5V}{32,767} \times \text{input value}^{\textcircled{1}} = \text{input voltage(V)}$$

^①The Input Value is the decimal value of the word in the input image for the corresponding analog input.

For example, if an input value of 16,021 is in the input image, the calculated value is:

$$\frac{10.5V}{32,767} \times 16,201 = 5.1915(V)$$

It should be noted that the actual value may vary within the accuracy limitations of the circuit.

To determine an approximate current that an input value represents, you can use the following equation:

$$\frac{21 \text{ mA}}{32,767} \times \text{input value}^{\textcircled{2}} = \text{input current (mA)}$$

[Ⓜ]The Input Value is the decimal value of the word in the input image for the corresponding analog input.

For example, if an input value of 4096 is in the input image, the calculated value is:

$$\frac{21 \text{ mA}}{32,767} \times 4096 = 2.625(\text{mA})$$

It should be noted that the actual value may vary within the accuracy limitations of the module.

Converting Analog Output Data

Use the following equation to determine the decimal value for the current output:

$$\frac{32,767}{16 \text{ mA}} \times [\text{Desired Current Output (mA)} - 4 \text{ mA}] = \text{Output Decimal Value}$$

For example, if an output value of 8 mA is desired, the value to be put in the corresponding word in the output image can be calculated as follows:

$$\frac{32,767}{16 \text{ mA}} \times (8 \text{ mA} - 4 \text{ mA}) = 8192$$

Use the following equation to determine the decimal value for the voltage output:

$$\frac{32,767}{10 \text{ V dc}} \times \text{Desired Voltage Output (V dc)} = \text{Output Decimal Value}$$

For example, if an output value of 1V dc is desired, the value to be put in the corresponding word in the output image can be calculated as follows:

$$\frac{32,767}{10 \text{ V dc}} \times 1 \text{ V dc} = 3277$$

Using Basic Instructions

This chapter contains general information about basic instructions and explains how they function in your application program. Each of the basic instructions includes information on:

- what the instruction symbol looks like
- typical execution time for the instruction
- how to use the instruction
- how to enter the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the basic instructions in use.

Bit Instructions

| HHP Display | Mnemonic | Function Code | Name | Purpose | Page |
|-------------|----------|---------------|--------------------------|--|------|
| ┌─┐ | LD | 20 | Load | Examines a bit for an On condition. It is the first normally open instruction in a rung or block. | 8-3 |
| ┌─/─┐ | LDI | 21 | Load Inverted | Examines a bit for an Off condition. It is the first normally closed instruction in a rung or block. | 8-4 |
| ─┐┐ | AND | 22 | And | Examines a bit for an On condition. It is a normally open instruction placed in series with any previous input instruction in the current rung or block. (This differs from the AND <i>output</i> instruction discussed in chapter 10.) | 8-3 |
| ─/─┐ | ANI | 23 | And Inverted | Examines a bit for an Off condition. It is a normally closed instruction placed in series with any previous input instruction in the current rung or block. | 8-4 |
| ┌─┐┐ | OR | 24 | Or | Examines a bit for an On condition. It is a normally open instruction placed in parallel with any previous input instruction in the current rung or block. (This differs from the OR <i>output</i> instruction discussed in chapter 11.) | 8-3 |
| ┌─/─┐┐ | ORI | 25 | Or Inverted | Examines a bit for an Off condition. It is a normally closed instruction placed in parallel with any previous input instruction in the current rung or block. | 8-4 |
| ┌─┐┐┐ | LDT | 26 | Load True | Represents a short as the first instruction in a rung or block. | 8-6 |
| ┌─┐┐┐┐ | ORT | 27 | Or True | Represents a short in parallel with the previous instruction in the current rung or block. | 8-6 |
| ┌─┐┐┐┐┐ | LD OSR | 28 | One-Shot Rising | Triggers a one-time event. | 8-7 |
| ─┐┐┐┐┐ | AND OSR | 29 | | | |
| ─ () ─ | OUT | 40 | Output (Output Energize) | Represents an output driven by some combination of input logic. Energized (1) when conditions preceding it permit power continuity in the rung and de-energized after the rung is false. | 8-8 |
| ─ (L) ─ | SET | 41 | Set (Output Latch) | Turns a bit on when the rung is executed and this bit retains its state when the rung is not executed or a power cycle occurs. | 8-8 |
| ─ (U) ─ | RST | 42 | Reset (Output Unlatch) | Turns a bit off when the rung is executed and this bit retains its state when the rung is not executed or when power cycle occurs. | 8-8 |

Continued on following page

Branch Instructions

| Mnemonic | Function Code | Name | Purpose | Page |
|----------|---------------|-------------|--|------|
| MPS | 10 | Memory Push | Stores the rung state that is present immediately preceding the MPS instruction. | 8-10 |
| MRD | 11 | Memory Read | Reads the rung state stored by the MPS instruction and resumes operation using that rung state. | 8-10 |
| MPP | 12 | Memory Pop | Removes the rung state stored by the MPS instruction, reads it, and resumes operation using that rung state. | 8-10 |
| ANB | 13 | And Block | Places two blocks of logic in series with each other (ANDs them). | 8-12 |
| ORB | 14 | Or Block | Places two blocks of logic in parallel with each other (ORs them). | 8-12 |

Timer/Counter Instructions

| Mnemonic | Function Code | Name | Purpose | Page |
|----------|---------------|-----------------|---|------|
| TON | 0 | Timer-On Delay | Counts timebase intervals when the instruction is true. | 8-16 |
| TOF | 1 | Timer-Off Delay | Counts timebase intervals when the instruction is false. | 8-18 |
| RTO | 2 | Retentive Timer | Counts timebase intervals when the instruction is true and retains the accumulated value when the instruction goes false or when power cycle occurs. | 8-20 |
| CTU | 5 | Count Up | Increments the accumulated value at each false-to-true transition and retains the accumulated value when the instruction goes false or when power cycle occurs. | 8-24 |
| CTD | 6 | Count Down | Decrements the accumulated value at each false-to-true transition and retains the accumulated value when the instruction goes false or when power cycle occurs. | 8-25 |
| RES | 7 | Reset | Resets the accumulated value and status bits of a timer or counter. Do not use with TOF timers. | 8-27 |

About Basic Instructions

Basic instructions are used most commonly for relay replacement functions, counting, and timing operations. These instructions, when used in Instruction List (Boolean) programs, represent hardwired logic circuits used for the control of a machine or equipment.

The basic instructions are separated into four groups: bit, branch, timer, and counter. Before you learn about the instructions in each of these groups, we suggest that you read the overview that precedes the group:

- Bit Instructions Overview
- Branch Instructions Overview
- Timer Instructions Overview
- Counter Instructions Overview

Bit Instructions Overview

These instructions operate on a single bit of data. During operation, the controller may set or reset the bit, based on the logical continuity of the rung. You can address a bit as many times as your program requires.

Important: Using the same address with multiple output instructions is not recommended.

Bit instructions are used with the following data files:

- Output (O) and input (I) data files. These represent external outputs and inputs.
- The status data file (S). These values are used to configure and provide controller status.
- The bit data file (B). These are the internal coils used in your program.
- Timer, counter, and control data files (T, C, and R). These instructions use various control bits.
- The integer data file (N). Use these addresses (at the bit level) as your program requires.

Load (LD), And (AND), and Or (OR)

Ladder representation:

—] [—

Execution Times (μsec) when:

| | True | False |
|-----|------|-------|
| LD | 1.54 | 1.72 |
| AND | 1.94 | 2.12 |
| OR | 1.94 | 2.12 |

LD, AND, and OR are all normally open instructions used in your program to determine if a bit is On. If the bit addressed is on (1) when one of these instructions is executed, then the instruction is evaluated as true. If the bit addressed is off (0) when one of these instructions is executed, then the instruction is evaluated as false. Refer to chapter 16 for instruction list examples of when to use the LD, AND, and OR instructions.

Important: The bit AND and bit OR instructions discussed here differ from the word AND and word OR *output* instructions described in chapter 11.

| Bit Address State | LD, AND, and OR Instruction |
|-------------------|-----------------------------|
| 0 | False |
| 1 | True |

Examples of devices that turn on or off include:

- a push button wired to an input (addressed as I/4)
- an output wired to a pilot light (addressed as O/2)
- a timer controlling a light (addressed as T3/DN)

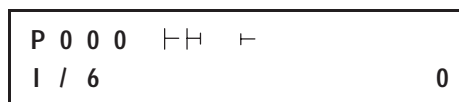
Using LD

Use the LD instruction for normally open contacts that appears first on a rung or block.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To access the LD instruction, press:



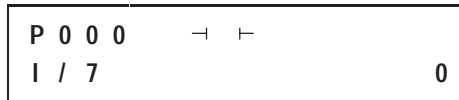
Using AND

Use the AND instruction for normally open contacts placed in series with any previous input instruction in the current rung or block.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To access the AND instruction, press:



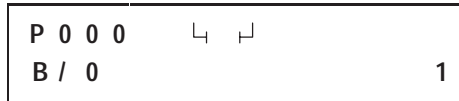
Using OR

Use the OR instruction for normally open contacts placed in parallel with any previous input instruction in the current rung or block.

Entering the Instruction

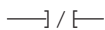
You enter the instruction from within the program monitor functional area.

To access the OR instruction, press:



Load Inverted (LDI), And Inverted (ANI), and Or Inverted (ORI)

Ladder representation:



Execution Times (μsec) when:

| | True | False |
|-----|------|-------|
| LDI | 1.54 | 1.72 |
| ANI | 1.94 | 2.12 |
| ORI | 1.94 | 2.12 |

LDI, ANI, and ORI are all normally closed instructions used in your ladder program to determine if a bit is Off. When the instruction is executed, if the bit addressed is off (0), then the instruction is evaluated as true. When the instruction is executed, if the bit addressed is on (1), then the instruction is evaluated as false.

| Bit Address State | LDI, ANI, and ORI Instruction |
|-------------------|-------------------------------|
| 0 | True |
| 1 | False |

Examples of devices that turn on or off include:

- motor overload normally closed (N.C.) wired to an input (I/10)
- an output wired to a pilot light (addressed as O/4)
- a timer controlling a light (addressed as T3/DN)

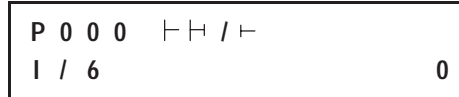
Using LDI

Use the LDI instruction for normally closed contacts that appears first on a rung or block.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To access the LDI instruction, press:



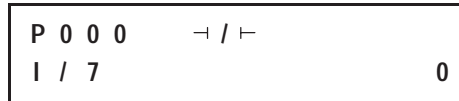
Using ANI

Use the ANI instruction for normally closed contacts placed in series with any previous input instruction in the current rung or block.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To access the ANI instruction, press:



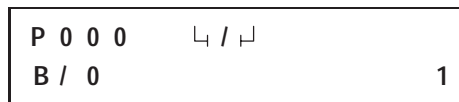
Using ORI

Use the ORI instruction for normally closed contacts placed in parallel with any previous input instruction in the current rung or block.

Entering the Instruction

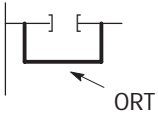
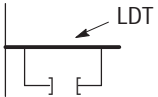
You enter the instruction from within the program monitor functional area.

To access the ORI instruction, press:



Load True (LDT) and Or True (ORT)

Ladder representation:



Execution Times (μsec) when:

| | True | False |
|-----|------|-------|
| LDT | 1.54 | n/a |
| ORT | 1.94 | n/a |

The LDT and ORT instructions are used to short circuit a block of logic. They are useful for debugging a rung when you need a condition that is always true.

There are no parameters to enter for these instructions.

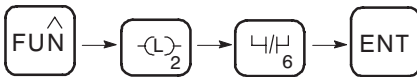
Using LDT

Use the LDT instruction to place a short in the first position of a rung or block. This instruction can only be used if it is immediately used with an OR or ORB instruction.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



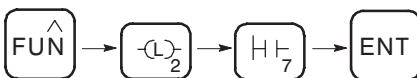
Using ORT

Use the ORT instruction to place a short in parallel with any previous input instruction in the current rung or block.

Entering the Instruction

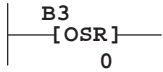
You enter the instruction from within the program monitor functional area.

To enter the function code, press:



One-Shot Rising (OSR)

Ladder representation:



Execution Times (μsec) when:

| | True | False |
|---------|-------|-------|
| LD OSR | 13.02 | 11.48 |
| AND OSR | 13.42 | 11.88 |

The OSR instruction is a retentive input instruction that triggers an event to occur one time. Use the OSR instruction when an event must start based on the change of state of the rung from false to true.

When the rung conditions preceding the OSR instruction go from false to true, the OSR instruction becomes true for one scan. After one scan is complete, the OSR instruction becomes false, even if the rung conditions preceding it remain true. The OSR instruction only becomes true again if the rung conditions preceding it transition from false to true.

The controller allows you to use one OSR instruction per output in a rung.

Entering Parameters

The address assigned to the OSR instruction is *not* the one-shot address referenced by your program, nor does it indicate the state of the OSR instruction. This address allows the OSR instruction to *remember* its previous rung state.

Use a bit address from either the bit or integer data file. The addressed bit is set (1) as long as rung conditions preceding the OSR instruction are true (even if the OSR instruction has become false); the bit is reset (0) when rung conditions preceding the OSR instruction are false.

Important: The bit address you use for this instruction must be unique. Do *not* use it elsewhere in the program.

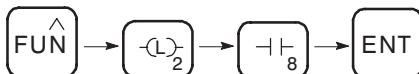
Do *not* use an input or output address to program the address parameter of the OSR instruction.

Entering the Instruction

| Mnemonic | HHP Display | Function Code | Use This Instruction When the Input: |
|----------|-------------|---------------|--|
| LD OSR | ┌OSR┐ | 28 | appears first on a rung or block |
| AND OSR | └OSR┘ | 29 | is placed in series with any previous input instruction in the current rung or block |

You enter the instruction from within the program monitor functional area. The example below shows how to access the LD OSR instruction. Use the same procedure to access the AND OSR instruction, only substitute the function code with the one provided in the table above.

To enter the function code, press:



Output (OUT)

Ladder representation:

—()—

Execution Times (μsec) when:

| True | False |
|------|-------|
| 4.43 | 4.43 |

Use an OUT instruction in your ladder program to turn On a bit when rung conditions are evaluated as true.

An example of a device that turns on or off is an output wired to a pilot light (addressed as O/4).

OUT instructions are reset when:

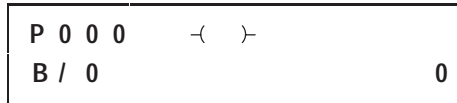
- You enter or return to the RRUN, RCSN, or RSSN mode or power is restored.
- The OUT is programmed within an inactive or false Master Control Reset (MCR) zone.
- Rung conditions are evaluated as false.

Important: A bit that is set within a subroutine using an OUT instruction remains set until the subroutine is scanned again.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To access the OUT instruction, press:



Set (SET) and Reset (RST)

Ladder representation:

—(L)—

—(U)—

Execution Times (μsec) when:

| | True | False |
|-----|------|-------|
| SET | 4.97 | 3.16 |
| RST | 4.97 | 3.16 |

SET and RST are retentive output instructions. SET can only turn on a bit, while RST can only turn off a bit. These instructions are usually used in pairs, with both instructions addressing the same bit.

Your program can examine a bit controlled by SET and RST instructions as often as necessary.



ATTENTION: Under fatal error conditions, physical outputs are turned off. Once the error conditions are cleared, the controller resumes operation using the data table value of the operand.

Using SET

When you assign an address to the SET instruction that corresponds to the address of a physical output, the output device wired to this screw terminal is energized when the bit is set (turned on or latched).

When rung conditions become false (after being true), the bit remains set and the corresponding output device remains energized.

When enabled, the SET instruction tells the controller to turn on the addressed bit. Thereafter, the bit remains on, regardless of the rung condition, until the bit is turned off (typically by a RST instruction in another rung).

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To access the SET instruction, press:



Using RST

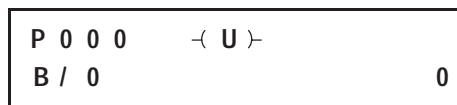
When you assign an address to the RST instruction that corresponds to the address of a physical output, the output device wired to this screw terminal is de-energized when the bit is cleared (turned off or unlatched).

The RST instruction tells the controller to turn off the addressed bit. Thereafter, the bit remains off, regardless of the rung condition, until it is turned on (typically by a SET instruction in another rung).

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To access the RST instruction, press:



Branch Instructions Overview

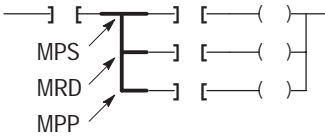
Branch instructions are connecting instructions unique to instruction list programming. There are two types of branch instructions:

- multiple output circuit connectors (MPS, MRD, and MPP)
- block connectors (ANB and ORB)

Since these instructions are used solely for connecting purposes, there are no parameters to enter.

Memory Push (MPS), Memory Read (MRD), and Memory Pop (MPP)

Ladder representation:



Execution Times (μsec) :

MPS 0.40
MRD 0.40
MPP 0.40

MPS, MRD, and MPP are multiple output circuit connecting instructions. These instructions work together to store, read, and clear the state of a rung prior to the execution of an output circuit.

Every MPS instruction used in a program must be paired with an MPP instruction. An MRD instruction is not always required.

Using MPS

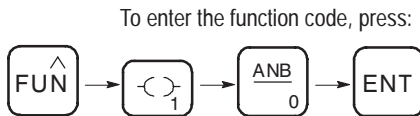
The MPS instruction stores the rung state immediately preceding the MPS instruction. This instruction is required only if *both* of the following statements are true:

- An output circuit that contains at least one input instruction immediately follows the MPS.
- There is at least one other output circuit after the output circuit that immediately follows the MPS.

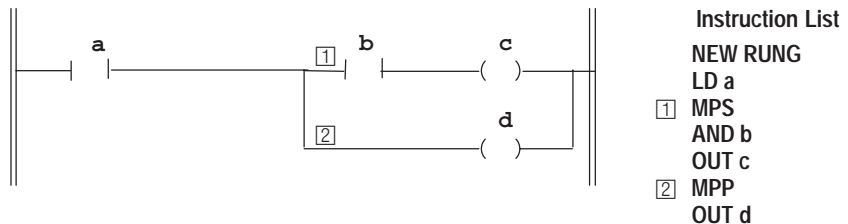
Every MPS instruction must be paired with an MPP instruction. Each rung can use a maximum of four nested MPS instructions.

Entering the Instruction

You enter the instruction from within the program monitor functional area.



The example below illustrates when you would enter the MPS instruction.



Using MRD

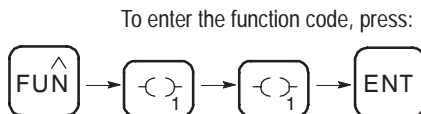
The MRD instruction reads the rung state stored by the MPS instruction and resumes operation using that rung state. Each branch structure can have a maximum of 73 MRD instructions.

This instruction can only be used if *all* of the following statements are true:

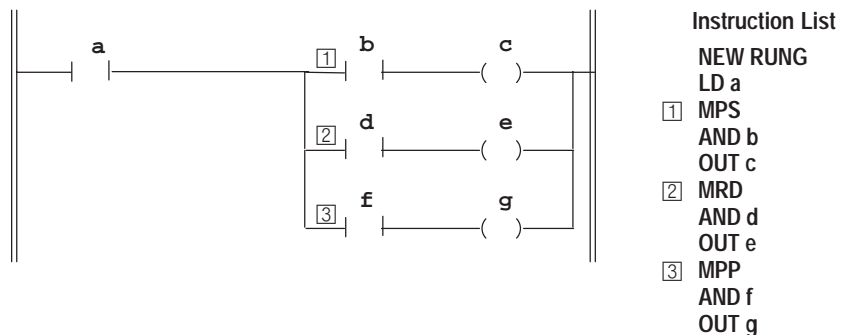
- An MPS instruction was used previously on the rung.
- The MRD is immediately preceded by an output instruction.
- An output circuit immediately follows the MRD.
- The output circuit that follows the MRD is not the last output circuit in the rung requiring the rung state stored by the MPS. (If it is the last output circuit requiring the stored rung state, you must use an MPP instruction instead of an MRD instruction.)

Entering the Instruction

You enter the instruction from within the program monitor functional area.



The example below illustrates when you would enter the MRD instruction.



Using MPP

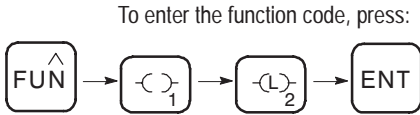
Like the MRD instruction, the MPP instruction reads the rung state stored by the MPS instruction and resumes operation using that rung state. Unlike the MRD instruction, the MPP also clears the rung state from the MPS.

This instruction can only be used if *all* of the following statements are true:

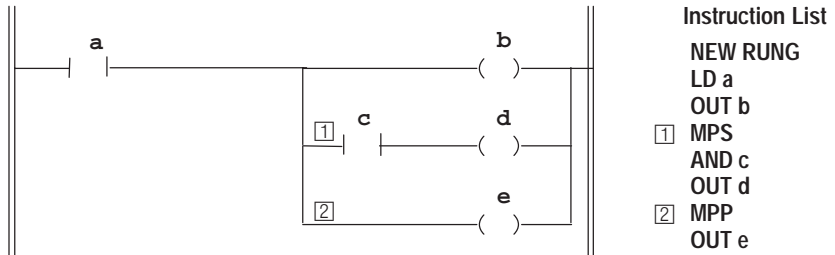
- An MPS instruction was used previously on the rung.
- The MPP is immediately preceded by an output instruction.
- An output circuit immediately follows the MPP.
- The output circuit that follows the MPP is the last output circuit in the rung requiring the rung state stored by the MPS. (If it is not the last output circuit requiring the stored rung state, you must use an MRD instruction instead of an MPP instruction.)

Entering the Instruction

You enter the instruction from within the program monitor functional area.

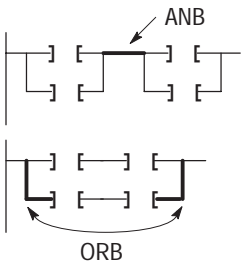


The example below illustrates when you would enter the MPP instruction.



And Block (ANB) and Or Block (ORB)

Ladder representation:



Execution Times (μsec):
ANB 0.40
ORB 0.40

ANB and ORB are block connecting instructions. These instructions connect two blocks of instructions in series or in parallel and produce a result that is used in the remaining execution of the rung.

Using ANB

The ANB instruction is used to make a series connection of circuit blocks with two or more contacts. (A series connection of circuit blocks with one contact requires only an AND or ANI instruction. See page 16–1.)

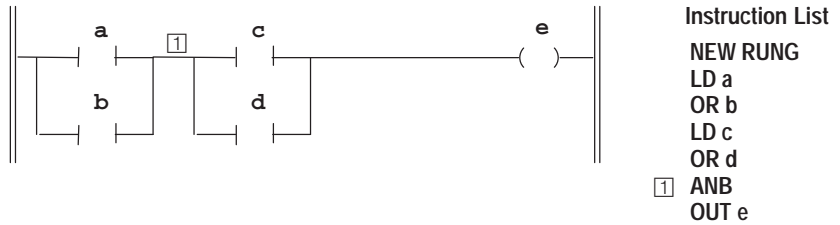
Entering the Instruction

You enter the instruction from within the program monitor functional area.

To access the ANB instruction, press:



The example below illustrates when you would enter the ANB instruction.



Using ORB

The ORB instruction is used to make a parallel connection of circuit blocks with two or more contacts. (A parallel connection of circuit blocks with one contact requires only an OR or ORI instruction. See page 16–2.)

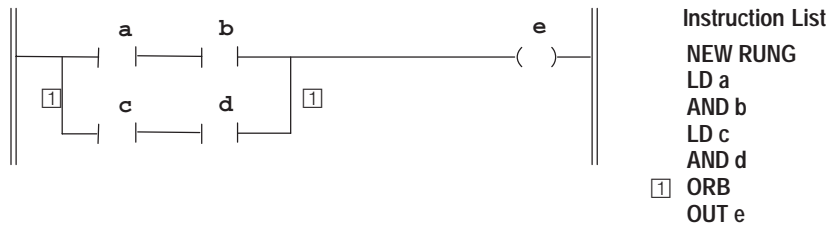
Entering the Instruction

You enter the instruction from within the program monitor functional area.

To access the ORB instruction, press:



The example below illustrates when you would enter the ORB instruction.



Timer Instructions Overview

Each timer address is made of a 3-word element. Word 0 is the control word, word 1 stores the preset value, and word 2 stores the accumulator value.

| | | |
|--------|-------------------|--------------|
| | 15 14 13 | |
| Word 0 | EN TT DN | Internal Use |
| Word 1 | Preset Value | |
| Word 2 | Accumulator Value | |

EN = Timer Enable Bit
 TT = Timer Timing Bit
 DN = Timer Done Bit

Entering Parameters

Accumulator Value (ACC)

This is the time elapsed since the timer was last reset. When enabled, the timer updates this continually.

Preset Value (PRE)

Specifies the value which the timer must reach before the controller sets the done bit. When the accumulated value becomes equal to or greater than the preset value, the done bit is set. You can use this bit to control an output device.

Preset and accumulated values for timers range from 0 to +32,767. If a timer preset or accumulated value is a negative number, a runtime error occurs.

Timebase

The timebase determines the duration of each timebase interval. The timebase is selectable as 0.01 (10 ms) second or 1.0 second.

Timer Accuracy

Timer accuracy refers to the length of time between the moment a timer instruction is enabled and the moment the timed interval is complete.

Timing accuracy is -0.01 to +0 seconds, with a program scan of up to 2.5 seconds. The 1-second timer maintains accuracy with a program scan of up to 1.5 seconds. If your programs can exceed 1.5 or 2.5 seconds, repeat the timer instruction rung so that the rung is scanned within these limits.

Important: Timing could be inaccurate if Jump (JMP), Label (LBL), Jump to Subroutine (JSR), or Subroutine (SBR) instructions skip over the rung containing a timer instruction while the timer is timing. If the skip duration is within 2.5 seconds, no time will be lost; if the skip duration exceeds 2.5 seconds, an undetectable timing error occurs. When using subroutines, a timer must be executed at least every 2.5 seconds to prevent a timing error.

Entering the Instructions

The following items apply when entering the instructions:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- If you see a down arrow on the display it means there are more options available. To scroll through the options, press this key:



- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. If you want to edit the instruction's parameters once the entire instruction is entered, you must go into the overwrite mode. (See page 17-4.)

Addressing Structure

Address bits and words using the format shown below.

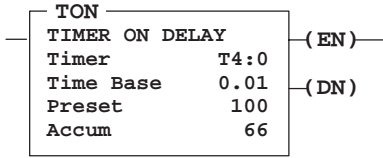
| Format | Explanation | |
|--------|-------------|---|
| Te.s/b | T | Timer file |
| | e | Element number Ranges from 0 – 39. These are 3-word elements. See figure on previous page. |
| | . | Subelement delimiter |
| | s | Subelement PRE or ACC |
| | / | Bit delimiter |
| | b | Bit number |

Addressing Examples

- **T0/EN** Enable bit
- **T0/TT** Timer timing bit
- **T0/DN** Done bit
- **T0.PRE** Preset value of the timer
- **T0.ACC** Accumulator value of the timer
- **T0.PRE/0** Bit 0 of the preset value
- **T0.ACC/0** Bit 0 of the accumulator value

Timer On-Delay (TON)

Ladder representation:



Execution Times (μsec) when:

| True | False |
|-------|-------|
| 38.34 | 30.38 |

Use the TON instruction to delay the turning on or off of an output. The TON instruction begins to count timebase intervals when rung conditions become true. As long as rung conditions remain true, the timer increments its accumulated value (ACC) each scan until it reaches the preset value (PRE). The accumulated value is reset when rung conditions go false, regardless of whether the timer has timed out.

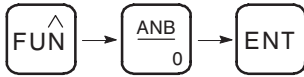
Using Status Bits

| This Bit | Is Set When | And Remains Set Until One of the Following |
|------------------------------|--|--|
| Timer Done Bit DN (bit 13) | accumulated value is equal to or greater than the preset value | rung conditions go false |
| Timer Enable Bit EN (bit 14) | rung conditions are true | rung conditions go false |
| Timer Timing Bit TT (bit 15) | rung conditions are true and the accumulated value is less than the preset value | rung conditions go false or when the done bit is set |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



```
P 0 0 0   T O N   A D D R
T 0
```

```
P 0 0 0   T O N   P R E
                * * * 1 0 0
```

```
P 0 0 0   T O N   A C C
                * * * 0 6 6
```

```
P 0 0 0   T O N   B A S E
0 . 0 1   S E C ↓
```

Once instruction entry is complete, the parameters are condensed to two screens as shown here:

| | | |
|---------|---------|-------|
| P 0 0 0 | T O N | T 0 0 |
| P | 1 0 0 A | 6 6 |

| | | |
|---------|-------|---------|
| P 0 0 0 | T O N | B A S E |
| 0 . 0 1 | S E C | |

When the controller changes from the RRUN, RCSN, or RSSN mode to the RPRG mode or user power is lost while the instruction is timing but has not reached its preset value, the following occurs:

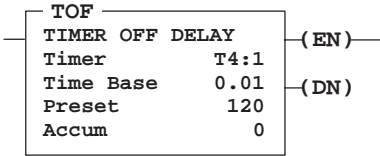
- Timer Enable (EN) bit remains set.
- Timer Timing (TT) bit remains set.
- Accumulated value (ACC) remains the same.

On returning to the RRUN, RCSN, or RSSN mode, the following can happen:

| Condition | Result |
|-----------------------|---|
| If the rung is true: | EN bit remains set. TT bit remains set. ACC value is reset. |
| If the rung is false: | EN bit is reset. TT bit is reset. ACC value is reset. |

Timer Off-Delay (TOF)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 39.42 | 31.65 |

Use the TOF instruction to delay turning on or off an output. The TOF instruction begins to count timebase intervals when the rung makes a true-to-false transition. As long as rung conditions remain false, the timer increments its accumulated value (ACC) each scan until it reaches the preset value (PRE). The controller resets the accumulated value when rung conditions go true regardless of whether the timer has timed out.

Using Status Bits

| This Bit | Is Set When | And Remains Set Until One of the Following |
|------------------------------|---|---|
| Timer Done Bit DN (bit 13) | rung conditions are true | rung conditions go false and the accumulated value is greater than or equal to the preset value |
| Timer Timing Bit TT (bit 14) | rung conditions are false and the accumulated value is less than the preset value | rung conditions go true or when the done bit is reset |
| Timer Enable Bit EN (bit 15) | rung conditions are true | rung conditions go false |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



```
P 0 0 0   T O F   A D D R
T 1
```

```
P 0 0 0   T O F   P R E
                * * * 1 2 0
```

```
P 0 0 0   T O F   A C C
                * * * * * 0
```

```
P 0 0 0   T O F   B A S E
0 . 0 1   S E C ↓
```

Once instruction entry is complete, the parameters are condensed to two screens as shown here:

| | | |
|---------|---------|-------|
| P 0 0 0 | T O F | T 0 1 |
| P | 1 2 0 A | 0 |

| | | |
|---------|-------|---------|
| P 0 0 0 | T O F | B A S E |
| 0 . 0 1 | S E C | |

When the controller changes from the RRUN, RCSN, or RSSN mode to the RPRG mode, or user power is lost while a timer off-delay instruction is timing but has not reached its preset value, the following occurs:

- Timer Enable (EN) bit remains set.
- Timer Timing (TT) bit remains set.
- Timer Done (DN) bit remains set.
- Accumulated value (ACC) remains the same.

On returning to the RRUN, RCSN, or RSSN mode, the following can happen:

| Condition | Result |
|-----------------------|---|
| If the rung is true: | TT bit is reset. DN bit remains set. EN bit is set. ACC value is reset. |
| If the rung is false: | TT bit is reset. DN bit is reset. EN bit is reset. ACC value is set equal to the preset value. |

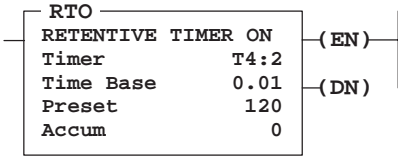


ATTENTION: The Reset (RES) instruction cannot be used with the TOF instruction because RES always clears the status bits as well as the accumulated value. (See page 8–27.)

Important: The TOF times inside an inactive MCR Pair.

Retentive Timer (RTO)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 38.34 | 27.49 |

Use the RTO instruction to turn an output on or off after its timer has been on for a preset time interval. The RTO instruction is a retentive instruction that lets the timer stop and start without resetting the accumulated value (ACC).

The RTO instruction retains its accumulated value when any of the following occurs:

- Rung conditions become false.
- You change controller operation from the RRUN, RCSN, or RSSN mode to the RPRG mode.
- The controller loses power.
- A fault occurs.

Using Status Bits

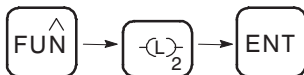
| This Bit | Is Set When | And Remains Set Until One of the Following |
|------------------------------|--|--|
| Timer Done Bit DN (bit 13) | accumulated value is equal to or greater than the preset value | the appropriate RES instruction is enabled |
| Timer Timing Bit TT (bit 14) | rung conditions are true and the accumulated value is less than the preset value | rung conditions go false or when the done bit is set |
| Timer Enable Bit EN (bit 15) | rung conditions are true | rung conditions go false |

Important: To reset the retentive timer's accumulated value and status bits after the RTO rung goes false, you must program a reset (RES) instruction with the same address in another rung.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



```

P 0 0 0   R T O   A D D R
T 2
  
```

```

P 0 0 0   R T O   P R E
                * * * 1 2 0
  
```

```

P 0 0 0   R T O   A C C
                * * * * * 0
  
```

```

P 0 0 0   R T O   B A S E
0 . 0 1   S E C ↓
  
```

Once instruction entry is complete, the parameters are condensed to two screens as shown here:

| | | |
|---------|---------|-------|
| P 0 0 0 | R T O | T 0 2 |
| P | 1 2 0 A | 0 |

| | | |
|---------|-------|---------|
| P 0 0 0 | R T O | B A S E |
| 0 . 0 1 | S E C | |

When the controller changes from the RRUN, RCSN, or RSSN mode to the RPRG or FLT mode or user power is lost while the timer is timing but not yet at the preset value, the following occurs:

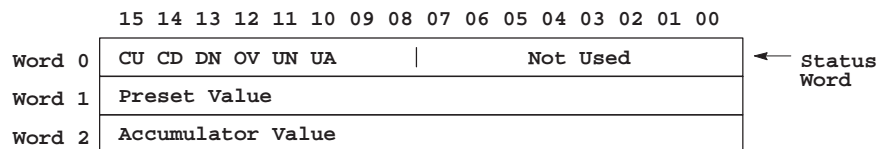
- Timer Enable (EN) bit remains set.
- Timer Timing (TT) bit remains set.
- Accumulated value (ACC) remains the same.

On returning to the RRUN, RCSN, or RSSN mode or when power is restored, the following can happen:

| Condition | Result |
|-----------------------|---|
| If the rung is true: | TT bit remains set. EN bit remains set. ACC value remains the same and resumes incrementing. |
| If the rung is false: | TT bit is reset. DN bit remains in its last state. EN bit is reset. ACC value remains in its last state. |

Counter Instructions Overview

Each Counter address is made of a 3-word data file element. Word 0 is the status word, containing six status bits. Word 1 is the preset value. Word 2 is the accumulated value.



CU = Counter Up Enable Bit
 CD = Counter Down Enable Bit
 DN = Done Bit
 OV = Overflow Occurred Bit
 UN = Underflow Occurred Bit
 UA = Update Counter Accumulator Bit

For high-speed counter instruction information, see chapter 14.

Entering Parameters

Accumulator Value (ACC)

This is the number of false-to-true transitions that have occurred since the counter was last reset.

Preset Value (PRE)

Specifies the value which the counter must reach before the controller sets the done bit. When the accumulator value becomes equal to or greater than the preset value, the done status bit is set. You can use this bit to control an output device.

Preset and accumulated values for counters range from $-32,768$ to $+32,767$, and are stored as signed integers. Negative values are stored in two's complement form.

Entering the Instructions

The following items apply when entering the instructions:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- If you see a down arrow on the display it means there are more options available. To scroll through the options, press this key:



- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters you must go into the overwrite mode. (See page 17-4.)

Addressing Structure

Address bits and words using the format shown below.

| Format | Explanation | |
|--------|-------------|---|
| Ce.s/b | C | Counter file |
| | e | Element number Ranges from 0 – 39. These are 3-word elements. See figure on page 8–21. |
| | . | Subelement delimiter |
| | s | Subelement PRE or ACC |
| | / | Bit delimiter |
| | b | Bit number |

Important: If assigned to a high-speed counter instruction, C0 is not available as an address for any other counter instructions. For more information on high-speed counter instructions, see chapter 14.

Addressing Examples

- **C0/CU** Count up enable bit
- **C0/CD** Count down enable bit
- **C0/DN** Done bit
- **C0/OV** Overflow bit
- **C0/UN** Underflow bit
- **C0/UA** Update accumulator bit

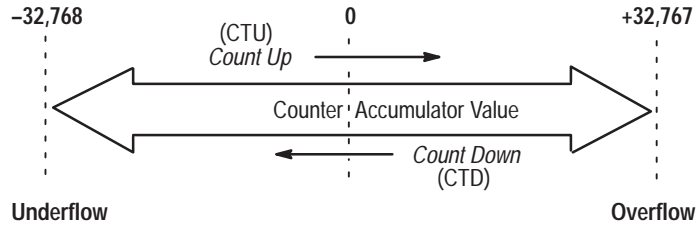
- **C0.PRE** Preset value of the counter
- **C0.ACC** Accumulator value of the counter

- **C0.PRE/0** Bit 0 of the preset value
- **C0.ACC/0** Bit 0 of the accumulated value

How Counters Work

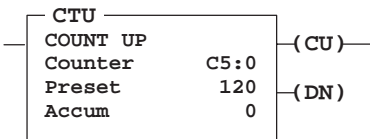
The figure below demonstrates how a counter works. The count value must remain in the range of $-32,768$ to $+32,767$. If the count value goes above $+32,767$ or below $-32,768$, a counter status overflow (OV) or underflow (UN) bit is set.

A counter can be reset to zero using the reset (RES) instruction. (See page 8–27.)



Count Up (CTU)

Ladder representation:



Execution Times (μ sec) when:

| True | False |
|-------|-------|
| 29.84 | 26.67 |

The CTU is an instruction that counts false-to-true rung transitions. Rung transitions can be caused by events occurring in the program (from internal logic or by external field devices) such as parts traveling past a detector or actuating a limit switch.

When rung conditions for a CTU instruction have made a false-to-true transition, the accumulated value is incremented by one count, provided that the rung containing the CTU instruction is evaluated between these transitions. The ability of the counter to detect false-to-true transitions depends on the speed (frequency) of the incoming signal.

Important: The on and off duration of an incoming signal must *not* be faster than the scan time $\times 2$ (assuming a 50% duty cycle).

The accumulated value is retained when the rung conditions again become false. The accumulated count is retained until cleared by a reset (RES) instruction that has the same address as the counter reset.

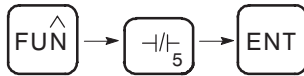
Using Status Bits

| This Bit | Is Set When | And Remains Set Until One of the Following |
|-----------------------------------|--|---|
| Count Up Overflow Bit OV (bit 12) | accumulated value wraps around to $-32,768$ (from $+32,767$) and continues counting up from there | a RES instruction having the same address as the CTU instruction is executed OR the count is decremented less than or equal to $+32,767$ with a CTD instruction |
| Done Bit DN (bit 13) | accumulated value is equal to or greater than the preset value | the accumulated value becomes less than the preset |
| Count Up Enable Bit EN (bit 15) | rung conditions are true | rung conditions go false OR a RES instruction having the same address as the CTU instruction is enabled |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



```
P 0 0 0   C T U   A D D R
C 0
```

```
P 0 0 0   C T U   P R E
                * * * 1 2 0
```

```
P 0 0 0   C T U   A C C
                * * * * * 0
```

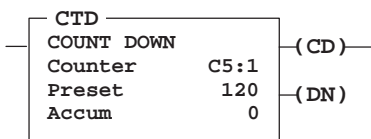
Once instruction entry is complete, the parameters are condensed to one screen as shown here:

```
P 0 0 0   C T U   C 0 0
P       1 2 0   A       0
```

The accumulated value is retained after the CTU instruction goes false or when power is removed from and then restored to the controller. Also, the on or off status of counter done, overflow, and underflow bits is retentive. The accumulated value and control bits are reset when the appropriate RES instruction is enabled. The CU bits are always set prior to entering the RRUN, RCSN, or RSSN modes.

Count Down (CTD)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 32.19 | 27.22 |

The CTD is a retentive output instruction that counts false-to-true rung transitions. Rung transitions can be caused by events occurring in the program such as parts traveling past a detector or actuating a limit switch.

When rung conditions for a CTD instruction have made a false-to-true transition, the accumulated value is decremented by one count, provided that the rung containing the CTD instruction is evaluated between these transitions.

The accumulated counts are retained when the rung conditions again become false. The accumulated count is retained until cleared by a reset (RES) instruction that has the same address as the counter reset.

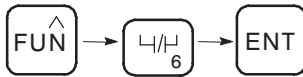
Using Status Bits

| This Bit | Is Set When | And Remains Set Until One of the Following |
|---|---|--|
| Count Down Underflow Bit UN (bit 11) | accumulated value wraps around to +32,768 (from -32,767) and continues counting down from there | a RES instruction having the same address as the CTD instruction is enabled. OR the count is incremented greater than or equal to +32,767 with a CTU instruction |
| Done Bit DN (bit 13) | accumulated value is equal to or greater than the preset value | the accumulated value becomes less than the preset |
| Count Down Enable Bit EN (bit 14) | rung conditions are true | rung conditions go false OR a RES instruction having the same address as the CTD instruction is enabled |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



```
P 0 0 0   C T D   A D D R
C 1
```

```
P 0 0 0   C T D   P R E
                * * * 1 2 0
```

```
P 0 0 0   C T D   A C C
                * * * * * 0
```

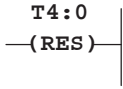
Once instruction entry is complete, the parameters are condensed to one screen as shown here:

```
P 0 0 0   C T D   C 0 1
P      1 2 0   A      0
```

The accumulated value is retained after the CTD instruction goes false or when power is removed from and then restored to the controller. Also, the on or off status of counter done, overflow, and underflow bits is retentive. The accumulated value and control bits are reset when the appropriate RES instruction is executed. The CD bits are always set prior to entering the RRUN, RCSN, or RSSN modes.

Reset (RES)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 15.19 | 4.25 |

Use a RES instruction to reset a timer or counter. When the RES instruction is executed, it resets the data having the same address as the RES instruction.

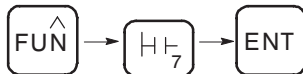
| Using a RES instruction for a: | The controller resets the: |
|---|--|
| Timer (Do not use a RES instruction with a TOF.) | ACC value to 0 DN bit TT bit EN bit |
| Counter | ACC value to 0 OV bit UN bit DN bit CU bit CD bit |
| Control | POS value to 0 EN bit EU bit DN bit EM bit ER bit UL bit IN and FD go to last state |

Important: If using this instruction to reset the high-speed counter accumulator, see page 14–19.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



| | |
|---------|-------|
| P 0 0 0 | R E S |
| T 0 0 | |

When resetting a counter, if the RES instruction is enabled and the counter rung is enabled, the CU or CD bit is reset.

If the counter preset value is negative, the RES instruction sets the accumulated value to zero. This in turn causes the done bit to be set by a count down or count up instruction.



ATTENTION: Because the RES instruction resets the accumulated value, and the done, timing, and enabled bits do *not* use the RES instruction to reset a timer address used in a TOF instruction. Otherwise, unpredictable machine operation or injury to personnel may occur.

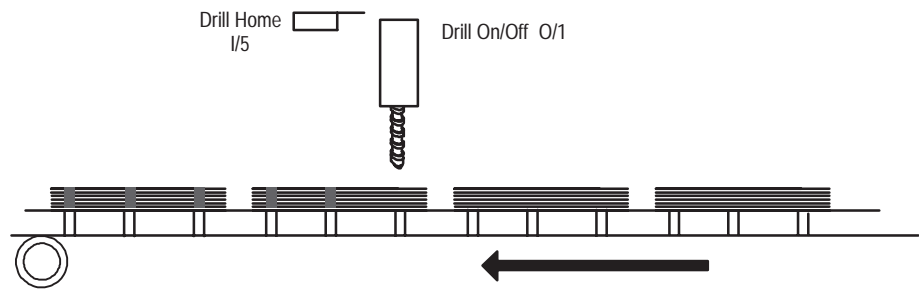
Basic Instructions in the Paper Drilling Machine Application Example

To demonstrate the use of basic instructions, this section provides ladder rungs followed by the optimized instruction list for these rungs. The rungs are part of the paper drilling machine application example described in appendix D. You will be updating the main program in file 2 and adding a subroutine to file 6.

Updating File 2

The rungs shown on the following page are referred to as the program's "start-up" logic. They determine the conditions necessary to start the machine in motion by monitoring the start and stop push buttons. When the start push button is pressed, it enables the conveyor to move and start spinning the drill bit. When the stop push button is pressed, it disables the conveyor motion and turns off the drill motor.

The start-up logic also checks to make sure that the drill is fully retracted (in the home position) before allowing the conveyor to move.

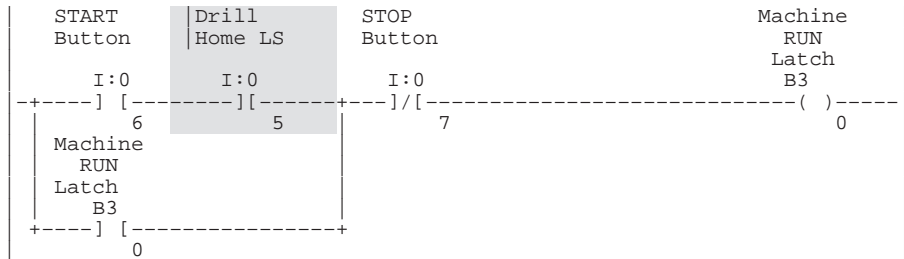


If you went through chapter 5, *Quick Start for New Users*, you already completed a portion of this example's main program file and saved it under the program name GETSTART. Adapt that program to the drilling example by adding the two shaded instructions.

Ladder Rungs

Rung 2:3^①

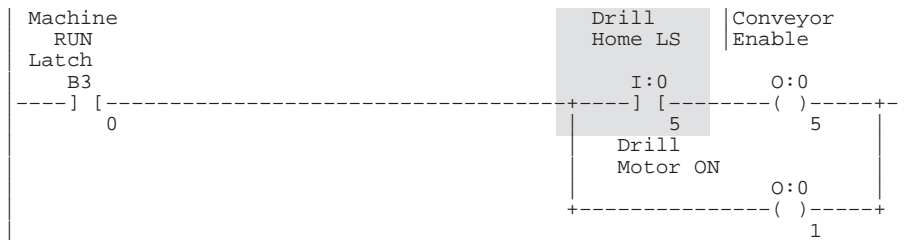
Starts the conveyor in motion when the start button is pressed. However, another condition must also be met before we start the conveyor: the drill must be in its fully retracted position (home). This rung also stops the conveyor when the stop button is pressed.



^① Rungs 2:0 through 2:2 will be added in chapter 14.

Rung 2:4

Applies the above start logic to the conveyor and drill motor.



Instruction List

File 2, Rung 3^①

Starts the conveyor in motion when the start button is pressed. However, another condition must also be met before we start the conveyor: the drill bit must be in its fully retracted position (home). This rung also stops the conveyor when the stop button is pressed.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME | ADDRESS | VALUE | FORCES |
|----------|----------------|----------|-------------------|---------|-------|--------|
| 20 | -] [- | LD | START Button | I/6 | 0 | |
| 22 | -] [- | AND | Drill Home LS | I/5 | 0 | |
| 24 | _] [_ | OR | Machine RUN Latch | B/0 | 0 | |
| 23 | -] / [- | ANI | STOP Button | I/7 | 0 | |
| 40 | - () - | OUT | Machine RUN Latch | B/0 | 0 | |

^① Rungs 2:0 through 2:2 will be added in chapter 14.

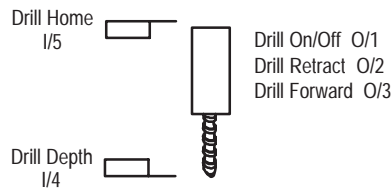
File 2, Rung 4

Applies the above start logic to the conveyor and drill motor.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| 20 | -] [- | LD | Machine RUN Latch B/0 | 0 | |
| 10 | | MPS | | | |
| 22 | -] [- | AND | Drill Home LS I/5 | 0 | |
| 12 | | MPP | | | |
| 40 | - () - | OUT | Conveyor Enable O/5 | 0 | |
| 40 | - () - | OUT | Drill Motor ON O/1 | 0 | |

Adding File 6

This subroutine controls the up and down motion of the drill for the paper drilling machine.



Ladder Rungs

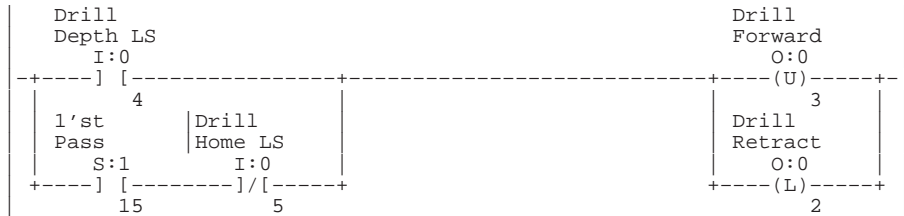
Rung 6:0

This section of ladder logic controls the up/down motion of the drill for the book drilling machine. When the conveyor positions the book under the drill, the DRILL SEQUENCE START bit is set. This rung uses that bit to begin the drilling operation. Because the bit is set for the entire drilling operation, the OSR is required to be able to turn off the forward signal so the drill can retract.



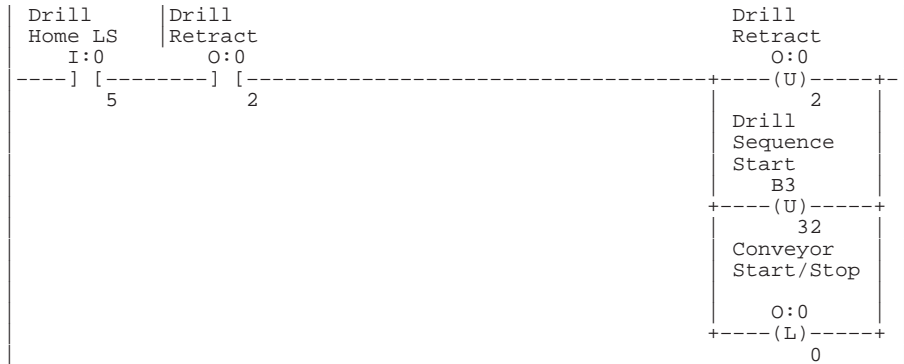
Rung 6:1

When the drill has drilled through the book, the body of the drill actuates the DRILL DEPTH limit switch. When this happens, the DRILL FORWARD signal is turned off and the DRILL RETRACT signal is turned on. The drill is also retracted automatically on power up if it is not actuating the DRILL HOME limit switch.

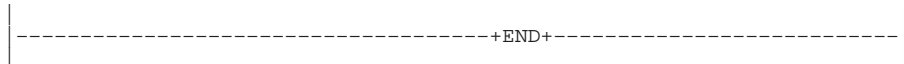


Rung 6:2

When the drill is retracting (after drilling a hole), the body of the drill actuates the DRILL HOME limit switch. When this happens the DRILL RETRACT signal is turned off, the DRILL SEQUENCE START bit is turned off to indicate the drilling process is complete, and the conveyor is restarted.



Rung 6.3



Instruction List

File 6, Rung 0

This section of ladder logic controls the up/down motion of the drill for the book drilling machine. When the conveyor positions the book under the drill, the DRILL SEQUENCE START bit is set. This rung uses that bit to begin the drilling operation. Because the bit is set for the entire drilling operation, the OSR is required to be able to turn off the forward signal so the drill can retract.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME | ADDRESS | VALUE | FORCES |
|----------|----------------|----------|----------------------|----------|-------|--------|
| 20 | -[] [- | LD | Drill Sequence Start | B/32 | 0 | |
| 29 | -OSR- | AND-OSR | Drill Subr | OSR B/48 | 0 | |
| 41 | -(L)- | SET | Drill Forward | O/3 | 0 | |

File 6, Rung 1

When the drill has drilled through the book, the body of the drill actuates the DRILL DEPTH limit switch. When this happens, the DRILL FORWARD signal is turned off and the DRILL RETRACT signal is turned on. The drill is also retracted automatically on power up if it is not actuating the DRILL HOME limit switch.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|----------------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -) [- | LD | Drill Depth LS | I/4 | 0 | |
| 20 | -) [- | LD | 1'st Pass | S1/15 | 0 | |
| 23 | -)/[- | ANI | Drill Home LS | I/5 | 0 | |
| 14 | | ORB | | | | |
| 42 | -(U)- | RST | Drill Forward | O/3 | 0 | |
| 41 | -(L)- | SET | Drill Retract | O/2 | 0 | |

File 6, Rung 2

When the drill is retracting (after drilling a hole), the body of the drill actuates the DRILL HOME limit switch. When this happens the DRILL RETRACT signal is turned off, the DRILL SEQUENCE START bit is turned off to indicate the drilling process is complete, and the conveyor is restarted.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|----------------------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -) [- | LD | Drill Home LS | I/5 | 0 | |
| 22 | -) [- | AND | Drill Retract | O/2 | 0 | |
| 42 | -(U)- | RST | Drill Retract | O/2 | 0 | |
| 42 | -(U)- | RST | Drill Sequence Start | B/32 | 0 | |
| 41 | -(L)- | SET | Conveyor Start/Stop | O/0 | 0 | |

Using Comparison Instructions

This chapter contains general information about comparison instructions and explains how they function in your application program. Each of the comparison instructions includes information on:

- what the instruction symbol looks like
- typical execution time for the instruction
- how to use the instruction
- how to enter the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the comparison instructions in use.

Comparison Instructions

| HHP Display | Mnemonic | Function Code | Name | Purpose | Page |
|-------------|----------|---------------|-----------------------|--|------|
| ┌EQU┐ | LD EQU | 50 | Equal | Test whether two values are equal. | 9-3 |
| └EQU┘ | AND EQU | 51 | | | |
| └EQU┘ | OR EQU | 52 | | | |
| ┌NEQ┐ | LD NEQ | 53 | Not Equal | Test whether one value is not equal to a second value. | 9-4 |
| └NEQ┘ | AND NEQ | 54 | | | |
| └NEQ┘ | OR NEQ | 55 | | | |
| ┌LES┐ | LD LES | 56 | Less Than | Test whether one value is less than a second value. | 9-5 |
| └LES┘ | AND LES | 57 | | | |
| └LES┘ | OR LES | 58 | | | |
| ┌LEQ┐ | LD LEQ | 59 | Less Than or Equal | Test whether one value is less than or equal to a second value. | 9-6 |
| └LEQ┘ | AND LEQ | 60 | | | |
| └LEQ┘ | OR LEQ | 61 | | | |
| ┌GRT┐ | LD GRT | 62 | Greater Than | Test whether one value is greater than another. | 9-7 |
| └GRT┘ | AND GRT | 63 | | | |
| └GRT┘ | OR GRT | 64 | | | |
| ┌GEQ┐ | LD GEQ | 65 | Greater Than or Equal | Test whether one value is greater than or equal to a second value. | 9-8 |
| └GEQ┘ | AND GEQ | 66 | | | |
| └GEQ┘ | OR GEQ | 67 | | | |

Continued on following page

| HHP Display | Mnemonic | Function Code | Name | Purpose | Page |
|-------------|----------|---------------|-----------------------------|---|------|
| ┌MEQ┐ | LD MEQ | 68 | Masked Comparison for Equal | Test portions of two values to see whether they are equal. Compares 16-bit data of a source address to 16-bit data at a reference address through a mask. | 9-9 |
| └MEQ┘ | AND MEQ | 69 | | | |
| └MEQ┘ | OR MEQ | 70 | | | |
| ┌LIM┐ | LD LIM | 71 | Limit Test | Test whether one value is within the limit range of two other values. | 9-10 |
| └LIM┘ | AND LIM | 72 | | | |
| └LIM┘ | OR LIM | 73 | | | |

About the Comparison Instructions

Comparison instructions are used to test pairs of values to condition the logical continuity of a rung. As an example, suppose a LES instruction is presented with two values. If the first value is less than the second, then the comparison instruction is true.

To learn more about the compare instructions, we suggest that you read the Compare Instructions Overview that follows.

Comparison Instructions Overview

The following general information applies to comparison instructions.

Entering the Instruction

The following items apply when entering the instructions:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters you must go into the overwrite mode. (See page 17-4.)

Function Codes

Each comparison instruction has three function codes associated with it. The code that you use correlates to the way the instruction is used on the rung, as described in the table below.

| If the instruction is | then use the code for |
|--|-----------------------|
| the first instruction on the rung or block | LD <instruction> |
| in series with another instruction | AND <instruction> |
| in parallel with another instruction | OR <instruction> |

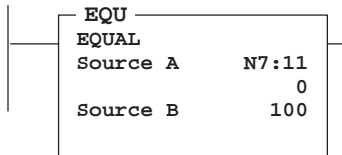
Since there are three codes for each instruction, you do not press a separate key to enter the action the instruction should take. (i.e., You do not press the LD instruction key and then enter a function code for EQU to get LD EQU.)

Using Indexed Word Addresses

When using comparison instructions, you have the option of using indexed word addresses for instruction parameters specifying word addresses. Indexed addressing is discussed in chapter 6.

Equal (EQU)

Ladder representation:



Execution Times (μsec) when:

| | True | False |
|---------|-------|-------|
| LD EQU | 21.52 | 6.60 |
| AND EQU | 21.92 | 7.00 |
| OR EQU | 21.92 | 7.00 |

Use the EQU instruction to test whether two values are equal. If source A and source B are equal, the instruction is logically true. If these values are not equal, the instruction is logically false.

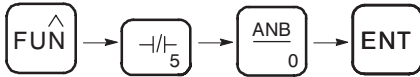
Source A must be a word address. Source B can be either a constant or word address. Negative integers are stored in two's complement form.

Entering the Instruction

| HHP Display | Mnemonic | Function Code | Use This Instruction When the Input: |
|-------------|----------|---------------|--|
| ┌EQU┐ | LD EQU | 50 | appears first on a rung or block |
| ─EQU─ | AND EQU | 51 | is placed in series with any previous input instruction in the current rung or block |
| └EQU┘ | OR EQU | 52 | is placed in parallel with any previous input instruction in the current rung or block |

You enter the instruction from within the program monitor functional area. The example that follows shows how to enter the LD EQU instruction. Use the same procedure to enter the other EQU instructions, only substitute the function code with one from the table above.

To enter the function code, press:

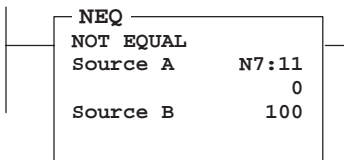


| | | |
|---------|---------|---------|
| P 0 0 0 | H E Q U | S R C A |
| N 1 1 | | 0 |

| | | |
|---------|---------|---------|
| P 0 0 0 | H E Q U | S R C B |
| | | 1 0 0 |

Not Equal (NEQ)

Ladder representation:



Use the NEQ instruction to test whether two values are not equal. If source A and source B are not equal, the instruction is logically true. If the two values are equal, the instruction is logically false.

Source A must be a word address. Source B can be either a constant or word address. Negative integers are stored in two's complement form.

Execution Times (μsec) when:

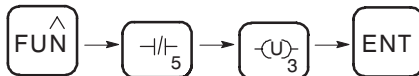
| | True | False |
|---------|-------|-------|
| LD NEQ | 21.52 | 6.60 |
| AND NEQ | 21.92 | 7.00 |
| OR NEQ | 21.92 | 7.00 |

Entering the Instruction

| HHP Display | Mnemonic | Function Code | Use This Instruction When the Input: |
|-------------|----------|---------------|--|
| HNEQ | LD NEQ | 53 | appears first on a rung or block |
| - HNEQ - | AND NEQ | 54 | is placed in series with any previous input instruction in the current rung or block |
| └ HNEQ ┘ | OR NEQ | 55 | is placed in parallel with any previous input instruction in the current rung or block |

You enter the instruction from within the program monitor functional area. The example below shows how to enter the LD NEQ instruction. Use the same procedure to enter the other NEQ instructions, only substitute the function code with one from the table above.

To enter the function code, press:

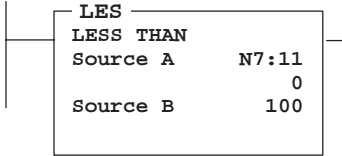


| | | |
|---------|---------|---------|
| P 0 0 0 | H N E Q | S R C A |
| N 1 1 | | 0 |

| | | |
|---------|---------|---------|
| P 0 0 0 | H N E Q | S R C B |
| | | 1 0 0 |

Less Than (LES)

Ladder representation:



Execution Times (µsec) when:

| | True | False |
|---------|-------|-------|
| LD LES | 23.60 | 6.60 |
| AND LES | 24.00 | 7.00 |
| OR LES | 24.00 | 7.00 |

Use the LES instruction to test whether one value (source A) is less than another (source B). If the value at source A is less than the value of source B the instruction is logically true. If the value at source A is greater than or equal to the value of source B, the instruction is logically false.

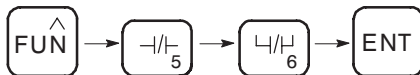
Source A must be a word address. Source B can be either a constant or word address. Negative integers are stored in two's complement form.

Entering the Instruction

| HHP Display | Mnemonic | Function Code | Use This Instruction When the Input: |
|-------------|----------|---------------|--|
| ┌LES┐ | LD LES | 56 | appears first on a rung or block |
| ¬LES┐ | AND LES | 57 | is placed in series with any previous input instruction in the current rung or block |
| ┌LES└┐ | OR LES | 58 | is placed in parallel with any previous input instruction in the current rung or block |

You enter the instruction from within the program monitor functional area. The example below shows how to enter the LD LES instruction. Use the same procedure to enter the other LES instructions, only substitute the function code with one from the table above.

To enter the function code, press:



```

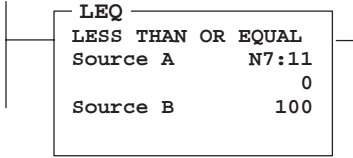
P 0 0 0   ┌LES┐   S R C A
N 1 1                                     0
    
```

```

P 0 0 0   ┌LES┐   S R C B
                                           1 0 0
    
```

Less Than or Equal (LEQ)

Ladder representation:



Execution Times (μsec) when:

| | True | False |
|---------|-------|-------|
| LD LEQ | 23.60 | 6.60 |
| AND LEQ | 24.00 | 7.00 |
| OR LEQ | 24.00 | 7.00 |

Use the LEQ instruction to test whether one value (source A) is less than or equal to another (source B). If the value at source A is less than or equal to the value of source B, the instruction is logically true. If the value at source A is greater than the value of source B, the instruction is logically false.

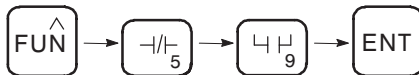
Source A must be a word address. Source B can be either a constant or word address. Negative integers are stored in two's complement form.

Entering the Instruction

| HHP Display | Mnemonic | Function Code | Use This Instruction When the Input: |
|-------------|----------|---------------|--|
| ┌LEQ┐ | LD LEQ | 59 | appears first on a rung or block |
| └LEQ┘ | AND LEQ | 60 | is placed in series with any previous input instruction in the current rung or block |
| └┌LEQ┐┘ | OR LEQ | 61 | is placed in parallel with any previous input instruction in the current rung or block |

You enter the instruction from within the program monitor functional area. The example below shows how to enter the LD LEQ instruction. Use the same procedure to enter the other LEQ instructions, only substitute the function code with one from the table above.

To enter the function code, press:



```

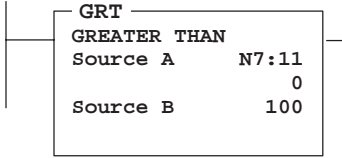
P 0 0 0  ┌LEQ┐  S R C A
N 1 1    0
    
```

```

P 0 0 0  ┌LEQ┐  S R C B
                1 0 0
    
```

Greater Than (GRT)

Ladder representation:



Execution Times (μsec) when:

| | True | False |
|---------|-------|-------|
| LD GRT | 23.60 | 6.60 |
| AND GRT | 24.00 | 7.00 |
| OR GRT | 24.00 | 7.00 |

Use the GRT instruction to test whether one value (source A) is greater than another (source B). If the value at source A is greater than the value of source B, the instruction is logically true. If the value at source A is less than or equal to the value of source B, the instruction is logically false.

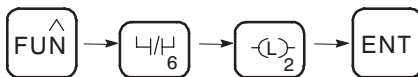
Source A must be a word address. Source B can be either a constant or word address. Negative integers are stored in two's complement form.

Entering the Instruction

| HHP Display | Mnemonic | Function Code | Use This Instruction When the Input: |
|-------------|----------|---------------|--|
| ┌GRT┐ | LD GRT | 62 | appears first on a rung or block |
| └GRT┘ | AND GRT | 63 | is placed in series with any previous input instruction in the current rung or block |
| └┌GRT┐┘ | OR GRT | 64 | is placed in parallel with any previous input instruction in the current rung or block |

You enter the instruction from within the program monitor functional area. The example below shows how to enter the LD GRT instruction. Use the same procedure to enter the other GRT instructions, only substitute the function code with one from the table above.

To enter the function code, press:



```

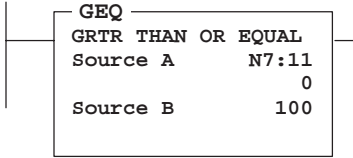
P 0 0 0   ┌ G R T ┐   S R C A
N 1 1                                     0
  
```

```

P 0 0 0   ┌ G R T ┐   S R C B
                                     1 0 0
  
```

Greater Than or Equal (GEQ)

Ladder representation:



Execution Times (µsec) when:

| | True | False |
|---------|-------|-------|
| LD GEQ | 23.60 | 6.60 |
| AND GEQ | 24.00 | 7.00 |
| OR GEQ | 24.00 | 7.00 |

Use the GEQ instruction to test whether one value (source A) is greater than or equal to another (source B). If the value at source A is greater than or equal to the value of source B, the instruction is logically true. If the value at source A is less than the value of source B, the instruction is logically false.

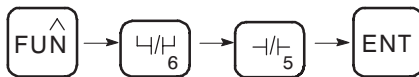
Source A must be a word address. Source B can be either a constant or word address. Negative integers are stored in two's complement form.

Entering the Instruction

| HHP Display | Mnemonic | Function Code | Use This Instruction When the Input: |
|-------------|----------|---------------|--|
| ┌GEQ┐ | LD GEQ | 65 | appears first on a rung or block |
| └GEQ┘ | AND GEQ | 66 | is placed in series with any previous input instruction in the current rung or block |
| └┌GEQ┐┘ | OR GEQ | 67 | is placed in parallel with any previous input instruction in the current rung or block |

You enter the instruction from within the program monitor functional area. The example below shows how to enter the LD GEQ instruction. Use the same procedure to enter the other GEQ instructions, only substitute the function code with one from the table above.

To enter the function code, press:



```

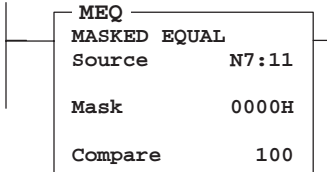
P 0 0 0  ┌ G E Q ┐  S R C A
N 1 1      0
    
```

```

P 0 0 0  ┌ G E Q ┐  S R C B
                1 0 0
    
```

Masked Comparison for Equal (MEQ)

Ladder representation:



Execution Times (μsec) when:

| | True | False |
|---------|-------|-------|
| LD MEQ | 28.39 | 7.69 |
| AND MEQ | 28.79 | 8.09 |
| OR MEQ | 28.79 | 8.09 |

Use the MEQ instruction to compare data of a source address with data of a reference address. Use of this instruction allows portions of the data to be masked by a separate word.

Entering Parameters

- **Source** is the address of the value you want to compare.
- **Mask** is the address of the mask through which the instruction moves data. The mask can be a hexadecimal value (constant).
- **Compare** is an integer value or the address of the reference.

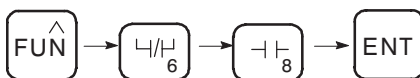
If the 16 bits of data at the source address are equal to the 16 bits of data at the compare address (less masked bits), the instruction is true. The instruction becomes false as soon as it detects a mismatch. Bits in the mask word mask data when reset; they pass data when set.

Entering the Instruction

| HHP Display | Mnemonic | Function Code | Use This Instruction When the Input: |
|-------------|----------|---------------|--|
| ┌MEQ┐ | LD MEQ | 68 | appears first on a rung or block |
| └MEQ┘ | AND MEQ | 69 | is placed in series with any previous input instruction in the current rung or block |
| └MEQ┐ | OR MEQ | 70 | is placed in parallel with any previous input instruction in the current rung or block |

You enter the instruction from within the program monitor functional area. The example below shows how to enter the LD MEQ instruction. Use the same procedure to enter the other MEQ instructions, only substitute the function code with one from the table above.

To enter the function code, press:



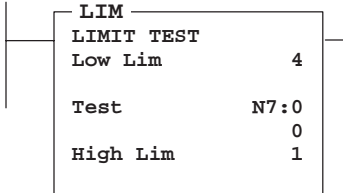
```
P 0 0 0  ┌ M E Q ┐  S R C
N 1 1                                0
```

```
P 0 0 0  ┌ M E Q ┐  M A S K
                                0 0 0 0 H
```

```
P 0 0 0  ┌ M E Q ┐  C O M P
                                1 0 0
```


Limit Test (LIM)

Ladder representation:



Execution Times (µsec) when:

| | True | False |
|---------|-------|-------|
| LD LIM | 36.93 | 7.69 |
| AND LIM | 37.33 | 8.09 |
| OR LIM | 37.33 | 8.09 |

Use the LIM instruction to test for values within or outside a specified range, depending on how you set the limits.

Entering Parameters

The Low Limit, Test, and High Limit values can be word addresses or constants, restricted to the following combinations:

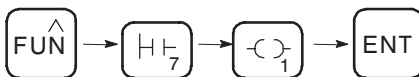
- If the Test parameter is a constant, both the Low Limit and High Limit parameters must be word addresses.
- If the Test parameter is a word address, the Low Limit and High Limit parameters can be either a constant or a word address.

Entering the Instruction

| HHP Display | Mnemonic | Function Code | Use This Instruction When the Input: |
|-------------|----------|---------------|--|
| ┌ LIM ─┐ | LD LIM | 71 | appears first on a rung or block |
| ─ LIM ─┐ | AND LIM | 72 | is placed in series with any previous input instruction in the current rung or block |
| ┌ LIM ─┐ | OR LIM | 73 | is placed in parallel with any previous input instruction in the current rung or block |

You enter the instruction from within the program monitor functional area. The example below shows how to enter the LD LIM instruction. Use the same procedure to enter the other LIM instructions, only substitute the function code with one from the table above.

To enter the function code, press:



```

P 0 0 0   ┌ L I M ─┐   L O W
                                     4
    
```

```

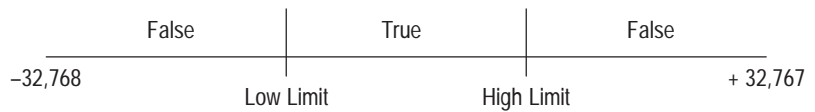
P 0 0 0   ┌ L I M ─┐   T E S T
N 0                                             0
    
```

```

P 0 0 0   ┌ L I M ─┐   H I G H
                                               1
    
```

True/False Status of the Instruction

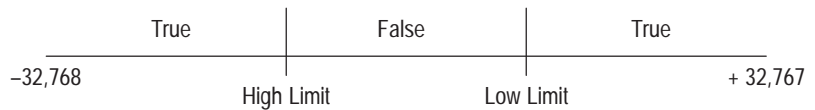
If the Low Limit has a value equal to or less than the High Limit, the instruction is true when the Test value is between the limits or is equal to either limit. If the Test value is outside the limits, the instruction is false, as shown below.



Example, low limit less than high limit:

| Low Limit | High Limit | Instruction is True when Test value is | Instruction is False when Test value is |
|-----------|------------|--|---|
| 5 | 8 | 5 through 8 | -32,768 through 4 and 9 through 32,767 |

If the Low Limit has a value greater than the High Limit, the instruction is false when the Test value is between the limits. If the Test value is equal to either limit or outside the limits, the instruction is true, as shown below.



Example, low limit greater than high limit:

| Low Limit | High Limit | Instruction is True when Test value is | Instruction is False when Test value is |
|-----------|------------|--|---|
| 8 | 5 | -32,768 through 5 and 8 through 32,767 | 6 and 7 |

Comparison Instructions in the Paper Drilling Machine Application Example

To demonstrate the use of comparison instructions, this section provides ladder rungs followed by the optimized instruction list for these rungs. The rungs are part of the paper drilling machine application example described in appendix D. You will be adding an instruction to file 2 and beginning a subroutine in file 7.

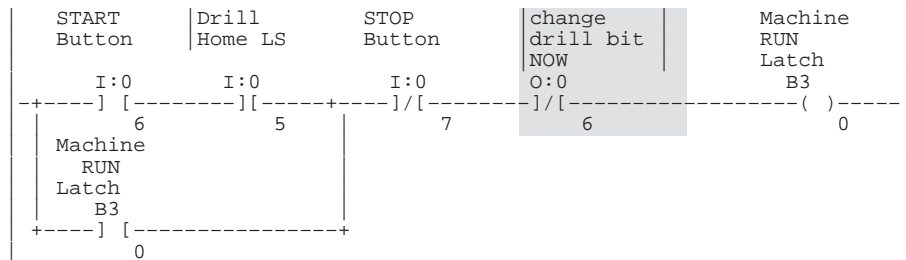
Adding to File 2

To begin you will once again need to return to the rungs first entered in chapter 5 in the program GETSTART. One more instruction needs to be added to the first rung to keep track of the drill life. This rung is indicated below by the shading. Notice that text has also been added to the rung comment.

Important: Do not add this instruction if you are using a 16 I/O controller. Address 00/6 is only valid for 32 I/O controllers.

Ladder Rung

Rung 2:3
Starts the conveyor in motion when the start button is pressed. However, there are other conditions that must also be met before we start the conveyor. They are: the drill must be in its fully retracted position (home); the drill bit must not be past its maximum useful life. This rung also stops the conveyor when the stop button is pressed or when the drill life is exceeded.



Instruction List

File 2, Rung 3

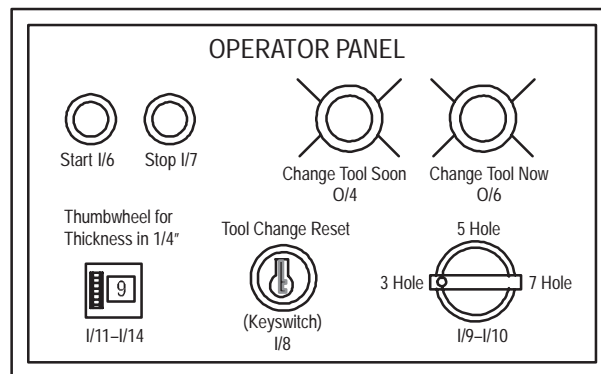
Starts the conveyor in motion when the start button is pressed. However, another condition must also be met before we start the conveyor: the drill bit must be in its fully retracted position (home). This rung also stops the conveyor when the stop button is pressed.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|-----------------------------|-------|--------|
| 20 | -] [- | LD | START Button I/6 | 0 | |
| 22 | -] [- | AND | Drill Home LS I/5 | 0 | |
| 24 | _] [_ | OR | Machine RUN Latch B/0 | 0 | |
| 23 | -] / [- | ANI | STOP Button I/7 | 0 | |
| 23 | -] / [- | ANI | change drill bit NOW O/6 | 0 | |
| 40 | - () - | OUT | Machine RUN Latch B/0 | 0 | |

Beginning a Subroutine in File 7

This section of ladder keeps track of the total inches of paper the current drill bit has drilled through. As the current bit wears out, a light illuminates on the operator panel, as shown below, to warn the operator to change the drill bit.

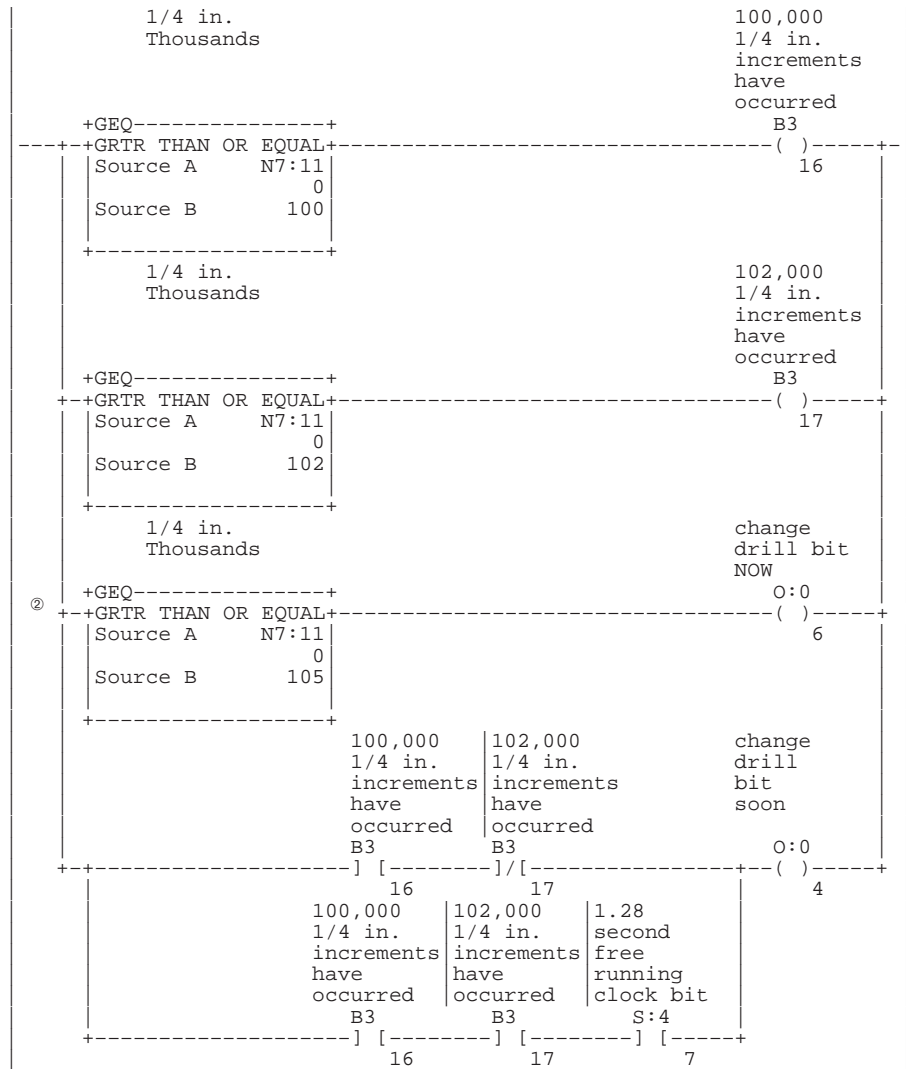
Note: For 32 I/O controllers: If the operator ignores this warning too long, this ladder shuts the machine down until the operator changes the bit.



Ladder Rung

Rung 7:0^①

Examines the number of 1/4 in. thousands that have accumulated over the life of the current drill bit. If the bit has drilled between 100,000-101,999 1/4 in. increments of paper, the "change drill" light illuminates steadily. When the value is between 102,000-103,999, the "change drill" light flashes at a 1.28 second rate. When the value reaches 105,000, the "change drill" light flashes, and the "change drill now" light illuminates.



① More rungs are added to this subroutine at the end of chapters 10 and 11.

② This branch accesses I/O only available with 32 I/O controllers. Therefore, do not include this branch if you are using a 16 I/O controller.

Instruction List

File 7, Rung 0^①

Examines the number of 1/4 in. thousands that have accumulated over the life of the current drill bit. If the bit has drilled between 100,000–101,999 1/4 in. increments of paper, the "change drill" light illuminates steadily. When the value is between 102,000–103,999, the "change drill" light flashes at a 1.28 second rate. When the value reaches 105,000, the "change drill" light flashes, and the "change drill now" light illuminates.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------------------|--|--------------|--------|
| ---- | ----- | ----- | ----- | ----- | ----- |
| 10 | | MPS | | | |
| 66 | -GEQ- | AND-GEQ | 1/4 in. Thousands SRCA N11 SRCB | 0000H 100 | |
| 40 | -()- | OUT | 100,000 1/4 in. increments have occurred B/16 | 0 | |
| 11 | | MRD | | | |
| 66 | -GEQ- | AND-GEQ | 1/4 in. Thousands SRCA N11 SRCB | 0000H 102 | |
| 40 | -()- | OUT | 102,000 1/4 in. increments have occurred B/17 | 0 | |
| 11 | | MRD | | | |
| 66 | -GEQ- | AND-GEQ ^② | 1/4 in. Thousands SRCA N11 SRCB | 0000H 105 | |
| 40 | -()- | OUT | change drill bit NOW O/6 | 0 | |
| 12 | | MPP | | | |
| 20 | -] [- | LD | 100,000 1/4 in. increments have occurred B/16 | 0 | |
| 23 | -] / [- | ANI | 102,000 1/4 in. increments have occurred B/17 | 0 | |
| 20 | -] [- | LD | 100,000 1/4 in. increments have occurred B/16 | 0 | |
| 22 | -] [- | AND | 102,000 1/4 in. increments have occurred B/17 | 0 | |
| 22 | -] [- | AND | 1.28 second free running clock bit S4/7 | 0 | |
| 14 | | ORB | | | |
| 13 | | ANB | | | |
| 40 | -()- | OUT | change drill bit soon O/4 | 0 | |

① More rungs are added to this subroutine at the end of chapters 10 and 11.

② This branch accesses I/O only available with 32 I/O controllers. Therefore, do not include this branch if you are using a 16 I/O controller.

Using Math Instructions

This chapter contains general information about math instructions and explains how they function in your logic program. Each of the math instructions includes information on:

- what the instruction symbol looks like
- typical execution time for the instruction
- how to use the instruction
- how to enter the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the math instructions in use.

Math Instructions

| Mnemonic | Function Code | Name | Purpose | Page |
|----------|---------------|---------------|---|-------|
| ADD | 80 | Add | Adds source A to source B and stores the result in the destination. | 10-4 |
| SUB | 81 | Subtract | Subtracts source B from source A and stores the result in the destination. | 10-5 |
| MUL | 82 | Multiply | Multiplies source A by source B and stores the result in the destination. | 10-8 |
| DIV | 83 | Divide | Divides source A by source B and stores the result in the destination and the math register. | 10-9 |
| DDV | 84 | Double Divide | Divides the contents of the math register by the source and stores the result in the destination and the math register. | 10-10 |
| CLR | 85 | Clear | Sets all bits of a word to zero. | 10-11 |
| SQR | 86 | Square Root | Calculates the square root of the source and places the integer result in the destination. | 10-11 |
| SCL | 87 | Scale Data | Multiplies the source by a specified rate, adds to an offset value, and stores the result in the destination. | 10-12 |

About the Math Instructions

These instructions perform the familiar four function math operations. The majority of the instructions take two input values, perform the specified arithmetic function, and output the result to an assigned memory location.

For example, both the ADD and SUB instructions take a pair of input values, add or subtract them, and place the result in the specified destination. If the result of the operation exceeds the allowable value, an overflow or underflow bit is set.

Since these are *output* instructions, they do not have LD, AND, and OR equivalents.

To learn more about the math instructions, we suggest that you read the Math Instructions Overview that follows.

Math Instructions Overview

The following general information applies to math instructions.

Entering the Instructions

The following items apply when entering the instructions:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters you must go into the overwrite mode. (See page 17–4.)

Using Indexed Word Addresses

You have the option of using indexed word addresses for instruction parameters specifying word addresses. Indexed addressing is discussed in chapter 6.

Updates to Arithmetic Status Bits

The arithmetic status bits are found in Word 0, bits 0–3 in the controller status file. After an instruction is executed, the arithmetic status bits in the status file are updated:

| With this Bit: | The Controller: |
|-------------------|--|
| S0/0 Carry (C) | sets if carry is generated; otherwise cleared. |
| S0/1 Overflow (V) | indicates that the actual result of a math instruction does not fit in the designated destination. |
| S0/2 Zero (Z) | indicates a 0 value after a math, move, or logic instruction. |
| S0/3 Sign (S) | indicates a negative (less than 0) value after a math, move, or logic instruction. |

Overflow Trap Bit, S5/0

Minor error bit (S5/0) is set upon detection of a mathematical overflow or division by zero. If this bit is set upon execution of an END statement or a Temporary End (TND) instruction, the recoverable major error code 0020 is declared.

In applications where a math overflow or divide by zero occurs, you can avoid a controller fault by using an reset (RST) instruction with address S5/0 in your program. The rung must be between the overflow point and the END or TND statement.

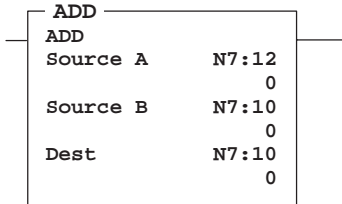
Changes to the Math Register, S13 and S14

Status word S13 contains the *least* significant word of the 32-bit values of the MUL and DDV instructions. It contains the remainder for DIV and DDV instructions. It also contains the first four BCD digits for the Convert from BCD (FRD) and Convert to BCD (TOD) instructions.

Status word S14 contains the *most* significant word of the 32-bit values of the MUL and DDV instructions. It contains the unrounded quotient for DIV and DDV instructions. It also contains the most significant digit (digit 5) for TOD and FRD instructions.

Add (ADD)

Ladder representation:



Execution Times (µsec) when:

| | |
|-------|-------|
| True | False |
| 33.09 | 6.78 |

Use the ADD instruction to add one value (source A) to another value (source B) and place the result in the destination.

Source A and B can either be a word address or a constant, however both sources cannot be a constant. The destination must be a word address.

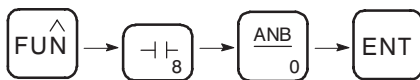
Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|--|
| S0/0 Carry (C) | sets if carry is generated; otherwise resets. |
| S0/1 Overflow (V) | sets if overflow is detected at destination; otherwise resets. On overflow, the minor error flag is also set. The value -32,768 or 32,767 is placed in the destination. If S2/14 (math overflow selection bit) is set, then the unsigned, truncated overflow remains in the destination. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets. |
| S0/3 Sign (S) | sets if result is negative; otherwise resets. |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



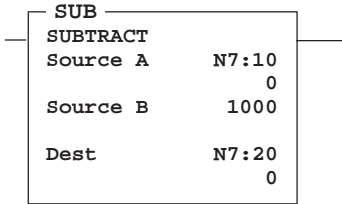
| | | |
|---------|-------|---------|
| P 0 0 0 | A D D | S R C A |
| N 1 2 | | 0 |

| | | |
|---------|-------|---------|
| P 0 0 0 | A D D | S R C B |
| N 1 0 | | 0 |

| | | |
|---------|-------|---------|
| P 0 0 0 | A D D | D E S T |
| N 1 0 | | 0 |

Subtract (SUB)

Ladder representation:



Execution Times (μsec) when:

| | |
|-------|-------|
| True | False |
| 33.52 | 6.78 |

Use the SUB instruction to subtract one value (Source B) from another (source A) and place the result in the destination.

Source A and B can either be a word address or a constant, however both sources cannot be a constant. The destination must be a word address.

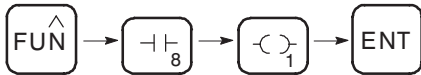
Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|--|
| S0/0 Carry (C) | sets if borrow is generated; otherwise resets. |
| S0/1 Overflow (V) | sets if underflow; otherwise reset. On underflow, the minor error flag is also set. The value -32,768 or 32,767 is placed in the destination. If S2/14 (math overflow selection bit) is set, then the unsigned, truncated overflow remains in the destination. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets. |
| S0/3 Sign (S) | sets if result is negative; otherwise resets. |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



| | | |
|---------|-------|---------|
| P 0 0 0 | S U B | S R C A |
| N 1 0 | | 0 |

| | | |
|---------|-------|---------|
| P 0 0 0 | S U B | S R C B |
| | | 1 0 0 0 |

| | | |
|---------|-------|---------|
| P 0 0 0 | S U B | D E S T |
| N 2 0 | | 0 |

32-Bit Addition and Subtraction

You have the option of performing 16-bit or 32-bit signed integer addition and subtraction. This is facilitated by status file bit S2/14 (math overflow selection bit).

Math Overflow Selection Bit S2/14

Set this bit when you intend to use 32-bit addition and subtraction. When S2/14 is set, and the result of an ADD, SUB, MUL, DIV, or NEG instruction cannot be represented in the destination address (due to math underflow or overflow):

- The overflow bit S0/1 is set.
- The overflow trap bit S5/0 is set.
- The destination address contains the unsigned truncated least significant 16 bits of the result.

When S2/14 is reset (default condition), and the result of an ADD, SUB, MUL, DIV, or NEG instruction cannot be represented in the destination address (due to math underflow or overflow):

- The overflow bit S0/1 is set.
- The overflow trap bit S5/0 is set.
- The destination address contains 32767 if the result is positive or -32768 if the result is negative.

Note that the status of bit S2/14 has no effect on the DDV instruction. Also, it has no effect on the math register content when using MUL and DIV instructions.

Example of 32-bit Addition

The following example shows how a 16-bit signed integer is added to a 32-bit signed integer. Remember that S2/14 must be set for 32-bit addition.

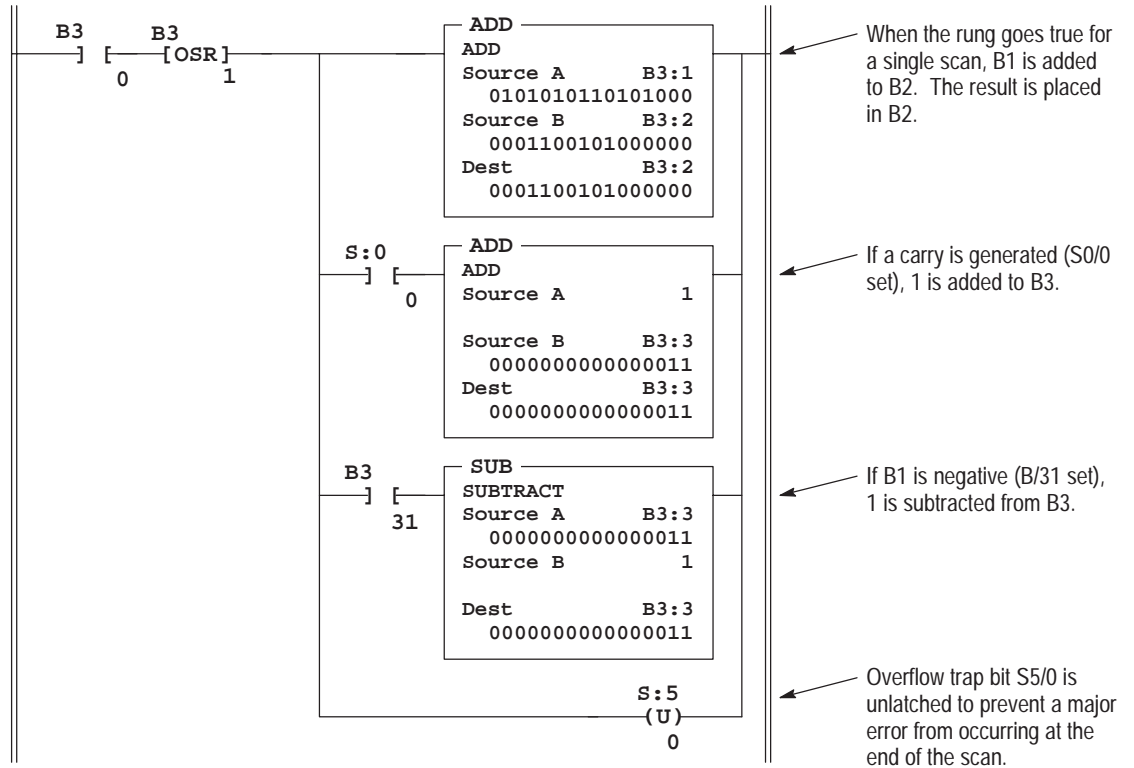
Note that the value of the most significant 16 bits (B3) of the 32-bit number is increased by 1 if the carry bit S0/0 is set and is decreased by 1 if the number being added (B1) is negative.

To avoid a major error from occurring at the end of the scan, you must unlatch overflow trap bit S5/0 as shown.

| Add 16-bit value B1 to 32-bit value B3 B2 | | | | | | | | | | | | | |
|---|--------|----|--------|------|------|------|------|------|------|------|------|----------------------|---------|
| Add Operation | | | Binary | | | | | | | | Hex | Decimal ^① | |
| Addend | B3 | B2 | 0000 | 0000 | 0000 | 0011 | 0001 | 1001 | 0100 | 0000 | 0003 | 1940 | 203,072 |
| | Addend | B1 | | | | | 0101 | 0101 | 1010 | 1000 | | 55A8 | 21,928 |
| Sum | B3 | B2 | 0000 | 0000 | 0000 | 0011 | 0110 | 1110 | 1110 | 1000 | 0003 | 6EE8 | 225,000 |

^① The programming device displays 16-bit decimal values only. The decimal value of a 32-bit integer is derived from the displayed binary or hex value. For example, 0003 1940 Hex is $16^4 \times 3 + 16^3 \times 1 + 16^2 \times 9 + 16^1 \times 4 + 16^0 \times 0 = 203,072$.

Ladder Rung

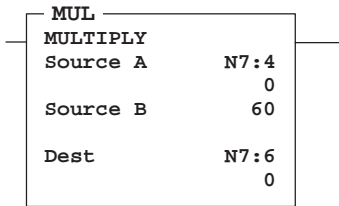


Instruction List

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|-------------------------------|------------------------------------|--------|
| 20 | [] [-] | LD | B/0 | 0 | |
| 29 | -OSR- | AND-OSR | B/1 | 0 | |
| 80 | | ADD | SRCA B1 SRCB B2 DEST B2 | 0101010110101000 1940H 1940H | |
| 10 | | MPS | | | |
| 22 | [] [-] | AND | S0/0 | 0 | |
| 80 | | ADD | SRCA SRCB B3 DEST B3 | 1 0003H 0003H | |
| 11 | | MRD | | | |
| 22 | [] [-] | AND | B/31 | 0 | |
| 81 | | SUB | SRCA B3 SRCB DEST B3 | 0003H 1 0003H | |
| 12 | | MPP | | | |
| 42 | -(U)- | RST | S5/0 | 0 | |

Multiply (MUL)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 57.96 | 6.78 |

Use the MUL instruction to multiply one value (source A) by another (source B) and place the result in the destination.

Source A and B can either be a word address or a constant, however both sources cannot be a constant. The destination must be a word address.

If the result is larger than +32,767 or smaller than -32,767 (16-bits), the 32-bit result is placed in the math register.

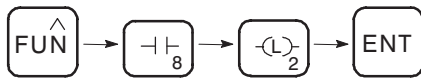
Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|--|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | sets if overflow is detected at destination; otherwise resets. On overflow, the minor error flag is also set. The value -32,768 or 32,767 is placed in the destination. If S2/14 (math overflow selection bit) is set, then the unsigned, truncated overflow remains in the destination. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets. |
| S0/3 Sign (S) | sets if result is negative; otherwise resets. |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



| | | |
|---------|-------|---------|
| P 0 0 0 | M U L | S R C A |
| N 4 | | 0 |

| | | |
|---------|-------|---------|
| P 0 0 0 | M U L | S R C B |
| | | 6 0 |

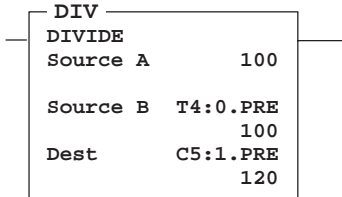
| | | |
|---------|-------|---------|
| P 0 0 0 | M U L | D E S T |
| N 6 | | 0 |

Changes to the Math Register

The math register contains the 32-bit signed integer result of the multiply operation. This result is valid at overflow.

Divide (DIV)

Ladder representation:



Execution Times (μsec) when:

| True | False |
|--------|-------|
| 147.87 | 6.78 |

Use the DIV instruction to divide one value (source A) by another (source B), and place the rounded quotient in the destination. If the remainder is 0.5 or greater, the destination is rounded up.

Source A and B can either be a word address or a constant; however, both sources cannot be a constant. The destination must be a word address.

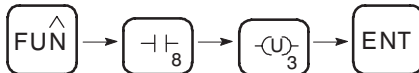
Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|--|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | sets if division by zero or overflow is detected; otherwise resets. On overflow, the minor error flag is also set. The value 32,767 is placed in the destination. If S2/14 (math overflow selection bit) is set, then the unsigned, truncated overflow remains in the destination. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets; undefined if overflow is set. |
| S0/3 Sign (S) | sets if result is negative; otherwise resets; undefined if overflow is set. |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



| | | |
|---------|-------|---------|
| P 0 0 0 | D I V | S R C A |
| | | 1 0 0 |

| | | |
|---------------|-------|---------|
| P 0 0 0 | D I V | S R C B |
| T 0 0 . P R E | | 1 0 0 |

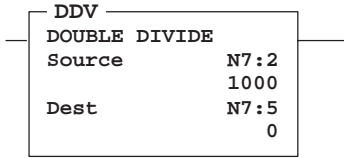
| | | |
|---------------|-------|---------|
| P 0 0 0 | D I V | D E S T |
| C 0 1 . P R E | | 1 2 0 |

Changes to the Math Register

The unrounded quotient is placed in the most significant word; the remainder is placed in the least significant word.

Double Divide (DDV)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|--------|-------|
| 157.06 | 6.78 |

The 32-bit content of the math register is divided by the 16-bit source value and the rounded quotient is placed in the destination. If the remainder is 0.5 or greater, the destination is rounded up.

The source can either be a word address or a constant. The destination must be a word address.

This instruction typically follows a MUL instruction that creates a 32-bit result.

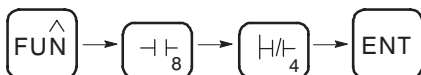
Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|--|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | sets if division by zero or if result is greater than 32,767 or less than -32,768; otherwise resets. On overflow, the minor error flag is also set. The value 32,767 is placed in the destination. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets. |
| S0/3 Sign (S) | sets if result is negative; otherwise resets; undefined if overflow is set. |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



| | | |
|---------|-------|---------|
| P 0 0 0 | D D V | S R C |
| N 2 | | 1 0 0 0 |

| | | |
|---------|-------|---------|
| P 0 0 0 | D D V | D E S T |
| N 5 | | 0 |

Changes to the Math Register

Upon instruction execution, the unrounded quotient is placed in the most significant word of the math register. The remainder is placed in the least significant word of the math register.

Clear (CLR)

Ladder representation:



Execution Times (μsec) when:

| True | False |
|-------|-------|
| 20.80 | 4.25 |

Use the CLR instruction to set the destination to zero. All of the bits reset. The destination must be a word address.

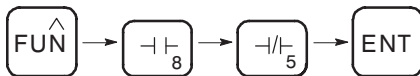
Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|-----------------|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | always resets. |
| S0/2 Zero (Z) | always sets. |
| S0/3 Sign (S) | always resets. |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

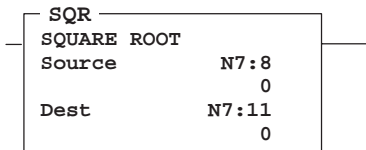
To enter the function code, press:



| | | |
|---------|-------|---------|
| P 0 0 0 | C L R | D E S T |
| N 1 1 | | 0 |

Square Root (SQR)

Ladder representation:



Execution Times (μsec) when:

| True | False |
|-------|-------|
| 71.25 | 6.78 |

When this instruction is evaluated as true, the square root of the absolute value of the source is calculated and the rounded integer result is placed in the destination. The source and the destination must be word addresses.

The instruction calculates the square root of a negative number without overflow or faults. In applications where the source value may be negative, use a comparison instruction to evaluate the source value to determine if the destination may be invalid.

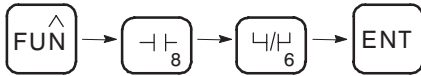
Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|--|
| S0/0 Carry (C) | sets if the source is negative; otherwise cleared. |
| S0/1 Overflow (V) | always resets. |
| S0/2 Zero (Z) | sets when destination value is zero. |
| S0/3 Sign (S) | always resets. |

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:

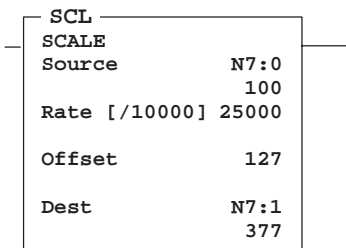


| | | |
|---------|-------|-------|
| P 0 0 0 | S Q R | S R C |
| N 8 | | 0 |

| | | |
|---------|-------|---------|
| P 0 0 0 | S Q R | D E S T |
| N 1 1 | | 0 |

Scale Data (SCL)

Ladder representation:



When this instruction is true, the value at the source address is multiplied by the rate value. The rounded result is added to the offset value and placed in the destination.

Important: Anytime an underflow or overflow occurs in the destination file, minor error bit S5/0 must be reset. This must occur before the end of the current scan to prevent major error code 0020 from being declared. This instruction can overflow before the offset is added.

Execution Times (µsec) when:

| True | False |
|--------|-------|
| 169.18 | 6.78 |

Entering Parameters

The value for the following parameters is between -32,768 to 32,767.

- **Source** must be a word address.
- **Rate** is the positive or negative value you enter divided by 10,000. It can be a constant or a word address.
- **Offset** can either be a constant or a word address.
- **Dest** must be a word address.

Updates to Arithmetic Status Bits

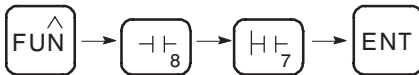
| With this Bit: | The Controller: |
|-------------------|---|
| S0/0 Carry (C) | is reserved. |
| S0/1 Overflow (V) | sets if an overflow is detected; otherwise resets. On overflow, minor error bit S:5/0 is also set and the value -32,768 or 32,767 is placed in the destination. The presence of an overflow is checked before and after the offset value is applied. ^① |
| S0/2 Zero (Z) | sets when destination value is zero. |
| S0/3 Sign (S) | sets if the destination value is negative; otherwise resets. |

^① If the result of the Source times the Rate, divided by 10000 is greater than 32,767, the SCL instruction overflows, causing error 0020 (Minor Error Bit), and places 32,767 in the Destination. This occurs regardless of the current offset.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

To enter the function code, press:



| | | |
|---------|-------|-------|
| P 0 0 0 | S C L | S R C |
| N 0 | | 1 0 0 |

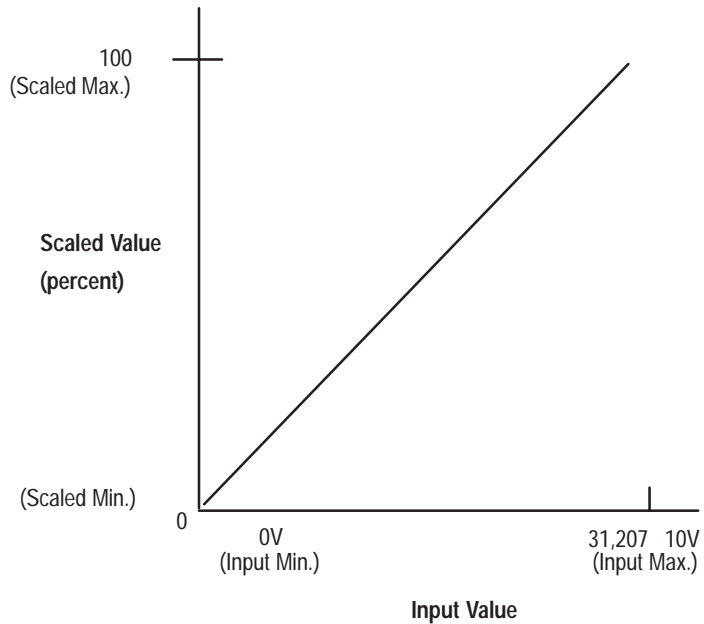
| | | |
|---------|-------|-----------|
| P 0 0 0 | S C L | R / 1 0 K |
| | | 2 5 0 0 0 |

| | | |
|---------|-------|---------|
| P 0 0 0 | S C L | O F S T |
| | | 1 2 7 |

| | | |
|---------|-------|---------|
| P 0 0 0 | S C L | D E S T |
| N 1 | | 3 7 7 |

The following example takes a 0V to 10.5V analog input from a MicroLogix 1000 analog controller and scales the raw input data to a value between 0 and 100%. The input value range is 0V to 10V which corresponds to 0 to 31,207 counts. The scaled value range is 0 to 100 percent.

Application Example – Convert Voltage Input to Percent



Calculating the Linear Relationship

Use the following equations to calculate the scaled units:

$$\text{Scaled value} = (\text{input value} \times \text{rate}) + \text{offset}$$

$$\text{Rate} = (\text{scaled max.} - \text{scaled min.}) / (\text{input max.} - \text{input min.})$$

$$= (100 - 0) / (31,207 - 0)$$

$$= .00320 \text{ (or } 32/10000)$$

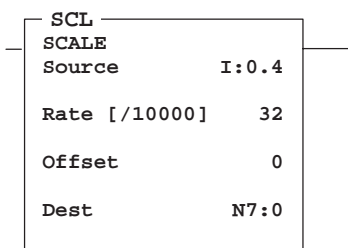
$$\text{Offset} = \text{scaled min.} - (\text{input min.} \times \text{rate})$$

$$= 0 - (0 \times .00320) = 0$$

As shown below, if the input value is 10, the scaled value is .032.

$$.032 = (10 \times .0032) + 0$$

Ladder representation:



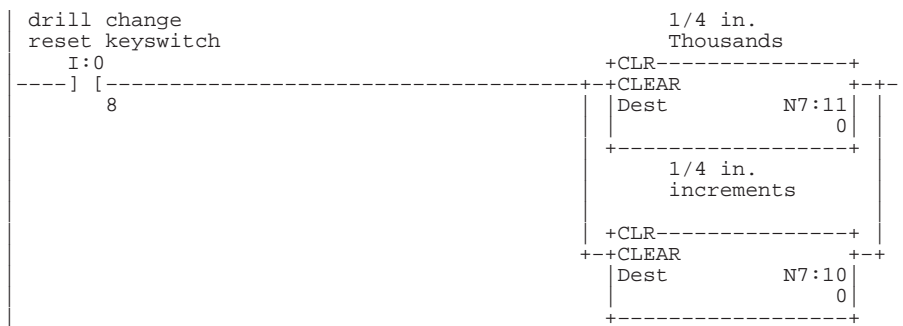
Math Instructions in the Paper Drilling Machine Application Example

To demonstrate the use of math instructions, this section provides ladder rungs followed by the optimized instruction list for these rungs. The rungs are part of the paper drilling machine application example described in appendix D. You will be adding to the subroutine in file 7 that was started in chapter 9.

Ladder Rungs

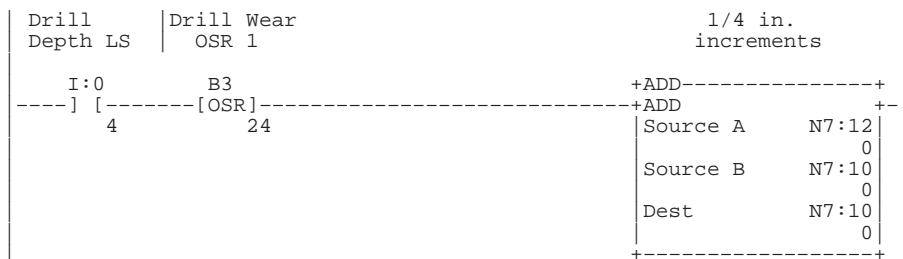
Rung 7:1

Resets the number of 1/4 in. increments and the 1/4 in. thousands when the "drill change reset" keyswitch is energized. This should occur following each drill bit change.



Rung 7:5^①

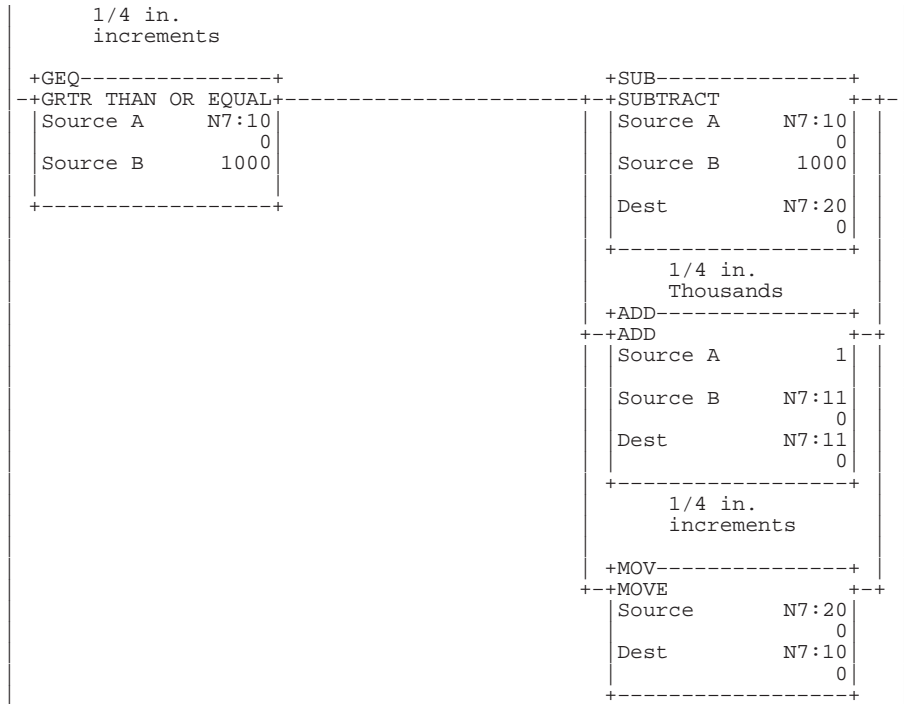
Keeps a running total of how many inches of paper have been drilled with the current drill bit. Every time a hole is drilled, the thickness (in 1/4 ins) is added to the running total (kept in 1/4 ins). The OSR is necessary because the ADD executes every time the rung is true, and the drill body would actuate the DRILL DEPTH limit switch for more than 1 program scan. Integer N7:12 is the integer-converted value of the BCD thumbwheel on inputs I:0/11 - I:0/14.



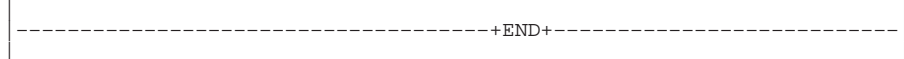
^① Rungs 7:2 through 7:4 are added at the end of Chapter 11.

Rung 7:6

When the number of 1/4 in. increments surpasses 1000, determines how many increments are past 1000 and stores in N7:20. Adds 1 to the total of 1000 1/4 in. increments and re-initializes the 1/4 in. increments accumulator to how many increments were beyond 1000.



Rung 7:7



Instruction List

File 7, Rung 1

Resets the number of 1/4 in. increments and the 1/4 in. thousands when the "drill change reset" keyswitch is energized. This should occur following each drill bit change.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--------------------------------|----------------|--------|
| 20 | -] [- | LD | drill change reset I/8 | keyswitch 0 | |
| 85 | | CLR | 1/4 in. Thousands DEST N11 | 0000H | |
| 85 | | CLR | 1/4 in. increments DEST N10 | 0000H | |

File 7, Rung 5^①

Keeps a running total of how many inches of paper have been drilled with the current drill bit. Every time a hole is drilled, adds the thickness (in 1/4 ins) to the running total (kept in 1/4 ins). The OSR is necessary because the ADD executes every time the rung is true, and the drill body would actuate the DRILL DEPTH limit switch for more than 1 program scan. Integer N7:12 is the integer-converted value of the BCD thumbwheel on inputs I:0/11 - I:0/14.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|-------------------------|--------|
| 20 | -] [- | LD | Drill Depth LS I/4 | 0 | |
| 29 | -OSR- | AND-OSR | Tool Wear OSR 1 B/24 | 0 | |
| 80 | | ADD | SRCA N12 1/4 in. increments SRCB N10 1/4 in. increments DEST N10 | 0000H 0000H 0000H | |

^① Rungs 7:2 through 7:4 are added at the end of Chapter 11.

File 7, Rung 6

When the number of 1/4 in. increments surpasses 1000, determines how many increments are past 1000 and stores in N7:20. Adds 1 to the total of '1000 1/4 in.' increments and re-initializes the 1/4 in. increments accumulator to how many increments were beyond 1000.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|------------------------|--------|
| 65 | - GEQ - | LD-GEQ | 1/4 in. increments SRCA N10 SRCB | 0000H 1000 | |
| 81 | | SUB | 1/4 in. increments SRCA N10 SRCB DEST N20 | 0000H 1000 0000H | |
| 80 | | ADD | SRCA 1/4 in. Thousands SRCB N11 1/4 in. Thousands DEST N11 | 1 0000H 0000H | |
| 106 | | MOV | SRC N20 1/4 in. increments DEST N10 | 0000H 0000H | |

Using Data Handling Instructions

This chapter contains general information about the data handling instructions and explains how they function in your application program. Each of the instructions includes information on:

- what the instruction symbol looks like
- typical execution time for the instruction
- how to use the instruction
- how to enter the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the data handling instructions in use.

Data Handling Instructions

| Mnemonic | Function Code | Name | Purpose | Page |
|----------|---------------|---------------------|--|-------|
| TOD | 100 | Convert to BCD | Converts the integer source value to BCD format and stores it in the destination. | 11-2 |
| FRD | 101 | Convert from BCD | Converts the BCD source value to an integer and stores it in the destination. | 11-3 |
| DCD | 102 | Decode 4 to 1 of 16 | Decodes a 4-bit value (0 to 15), turning on the corresponding bit in the 16-bit destination. | 11-7 |
| ENC | 103 | Encode 1 of 16 to 4 | Encodes a 16-bit source to a 4-bit value. Searches the source from the lowest to the highest bit, and looks for the first set bit. The corresponding bit position is written to the destination as an integer. | 11-8 |
| COP | 104 | Copy File | The COP instruction copies data from the source file to the destination file. The FLL instruction loads a source value into each position in the destination file. | 11-10 |
| FLL | 105 | Fill File | | |
| MOV | 106 | Move | Moves the source value to the destination. | 11-15 |
| MVM | 107 | Masked Move | Moves data from a source location to a selected portion of the destination. | 11-16 |
| AND | 108 | And | Performs a bitwise AND operation. (This differs from the AND <i>input</i> instruction discussed in chapter 7.) | 11-18 |
| OR | 109 | Or | Performs a bitwise inclusive OR operation. (This differs from the OR <i>input</i> instruction discussed in chapter 7.) | 11-19 |
| XOR | 110 | Exclusive Or | Performs a bitwise Exclusive OR operation. | 11-20 |
| NOT | 111 | Not | Performs a NOT operation. | 11-21 |
| NEG | 112 | Negate | Changes the sign of the source and stores it in the destination. | 11-22 |
| FFL | 113 | FIFO Load | The FFL instruction loads a word into a FIFO stack on successive false-to-true transitions. The FFU unloads a word from the stack on successive false-true transitions. The first word loaded is the first to be unloaded. | 11-25 |
| FFU | 114 | FIFO Unload | | |
| LFL | 115 | LIFO Load | The LFL instruction loads a word into a LIFO stack on successive false-to-true transitions. The LFU unloads a word from the stack on successive false-to-true transitions. The last word loaded is the first to be unloaded. | 11-28 |
| LFU | 116 | LIFO Unload | | |

About the Data Handling Instructions

Use these instructions to convert information, manipulate data in the controller, and perform logic operations.

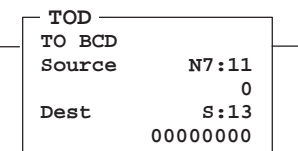
Since these are *output* instructions, they do not have LD, AND, and OR equivalents.

In this chapter you will find a general overview preceding groups of instructions. Before you learn about the instructions in each of these groups, we suggest that you read the overview. This chapter contains the following overviews:

- Move and Logical Instructions Overview
- FIFO and LIFO Instructions Overview

Convert to BCD (TOD)

Ladder representation:



Execution Times (μsec) when:

| True | False |
|-------|-------|
| 49.64 | 6.78 |

Use this instruction to convert 16-bit integers into BCD values.

The source must be a word address. The destination parameter can be a word address in a data file, or it can be the math register, S13 and S14.

If the integer value you enter is negative, the sign is ignored and the conversion occurs as if the number was positive.

Updates to Arithmetic Status Bits

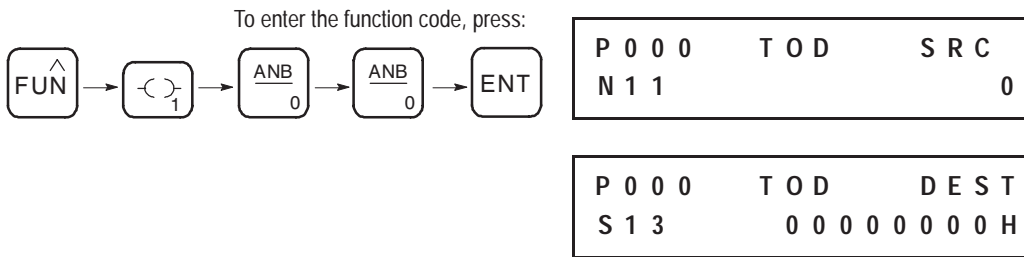
| With this Bit: | The Controller: |
|-------------------|--|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | sets if the BCD result is larger than 9999. On overflow, the minor error flag is also set. |
| S0/2 Zero (Z) | sets if destination value is zero. |
| S0/3 Sign (S) | sets if the source word is negative; otherwise resets. |

Entering the Instruction

You enter the instruction from within the program monitor functional area. As you enter the instruction, you can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters you must go into the overwrite mode. (See page 17–4.)



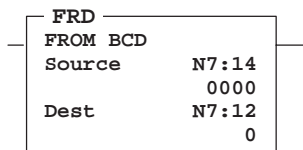
Changes to the Math Register

Contains the 5-digit BCD result of the conversion. This result is valid at overflow.

Important: To convert numbers larger than 9999 decimal, the destination must be the Math Register (S13). You must reset the Minor Error Bit (S5/0) to prevent an error.

Convert from BCD (FRD)

Ladder representation:



Execution Times (µsec) when:

| | |
|-------|-------|
| True | False |
| 56.88 | 5.52 |

Use this instruction to convert BCD values to integer values.

The source parameter can be a word address in a data file, or it can be the math register, S13. The destination must be a word address.

Updates to Arithmetic Status Bits

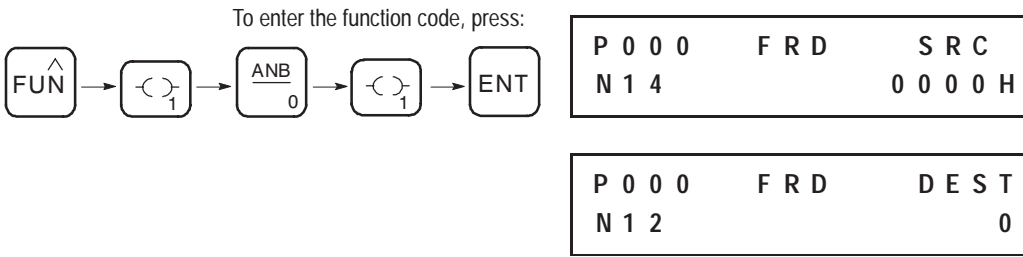
| With this Bit: | The Controller: |
|----------------------|---|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | sets if non-BCD value is contained at the source or the value to be converted is greater than 32,767; otherwise reset. On overflow, the minor error flag is also set. |
| S0/2 Zero (Z) | sets if destination value is zero. |
| S0/3 Sign (S) | always resets. |

Entering the Instruction

You enter the instruction from within the program monitor functional area. As you enter the instruction, you can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters you must go into the overwrite mode. (See page 17-4.)



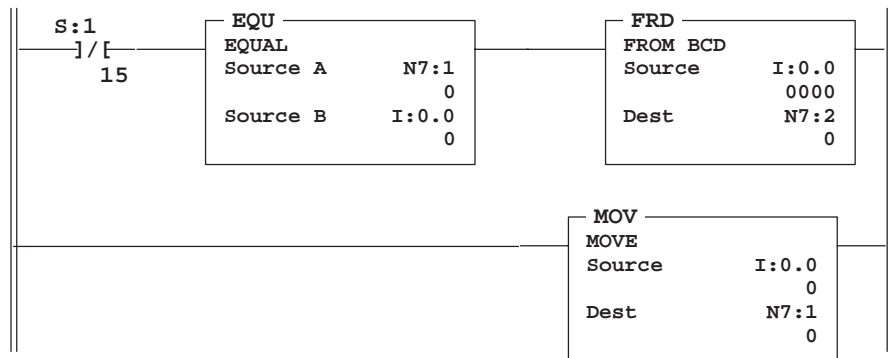
Important: Always provide filtering of all BCD input devices prior to performing the FRD instruction. The slightest difference in point-to-point input filter delay can cause the FRD instruction to overflow due to the conversion of a non-BCD digit.

Example 1

In the following example, the two rungs cause the controller to verify that the value I0 remains the same for two consecutive scans before it executes the FRD. This prevents the FRD from converting a non-BCD value during an input value change.

Important: To convert numbers larger than 9999 BCD, the source must be the Math Register (S13). You must reset the Minor Error Bit (S5/0) to prevent an error.

Ladder Rungs



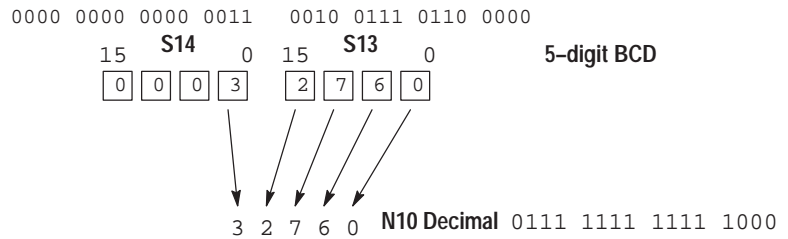
Instruction List

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|----------------|--------|
| 21 | -]/[- | LDI | S1/15 | 0 | ----- |
| 51 | -EQU- | AND-EQU | SRCA N1 SRCB I0 | 0000H 0000H | |
| 101 | | FRD | SRC I0 DEST N2 | 0000H 0000H | |

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|----------------|--------|
| 106 | | MOV | SRC I0 DEST N1 | 0000H 0000H | ----- |

Example 2

The BCD value 32760 in the math register is converted and stored in N10. The maximum source value is 32767, BCD.

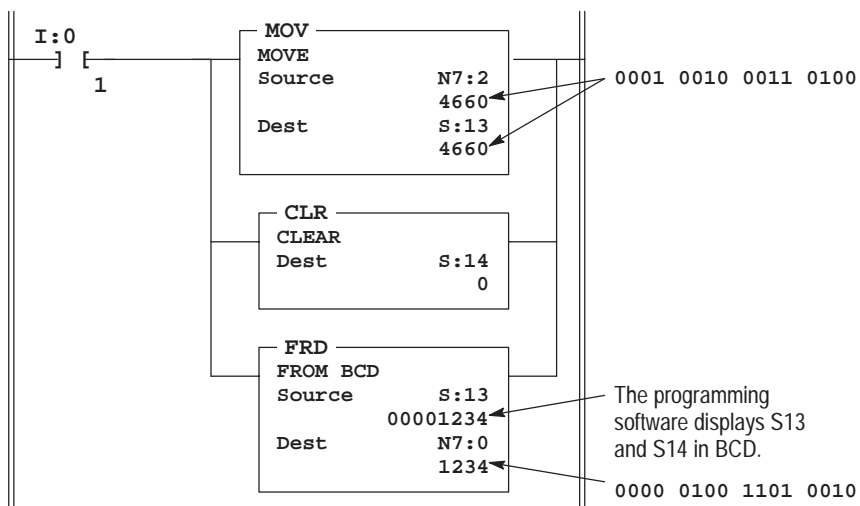


You should convert BCD values to integer before you manipulate them in your program. If you do not convert the values, the controller manipulates them as integers and their value may be lost.

Important: If the math register (S13 and S14) is used as the source for the FRD instruction and the BCD value does not exceed 4 digits, be sure to clear word S14 before executing the FRD instruction. If S14 is not cleared and a value is contained in this word from another math instruction located elsewhere in the program, an incorrect decimal value will be placed in the destination word.

Clearing S14 before executing the FRD instruction is shown below. When the input condition I/1 is set (1), a BCD value (transferred from a 4-digit thumbwheel switch for example) is moved from word N2 into the math register. Status word S14 is then cleared to make certain that unwanted data is not present when the FRD instruction is executed.

Ladder Rung

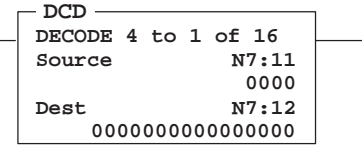


Instruction List

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------------------|--------|
| 20 | -] [- | LD | I/1 | 0 | |
| 106 | | MOV | SRC N2 DEST S13 | 4660 4660 | |
| 85 | | CLR | DEST S14 | 0000H | |
| 101 | | FRD | SRC S13 DEST N0 | 00001234H 1234 | |

Decode 4 to 1 of 16 (DCD)

Ladder representation:



Execution Times (µsec) when:

| | |
|-------|-------|
| True | False |
| 27.67 | 6.78 |

When executed, this instruction sets one bit of the destination word. The particular bit that is turned On depends on the value of the first four bits of the source word. See the table below.

Use this instruction to multiplex data in applications such as rotary switches, keypads, and bank switching.

| Bit | Source | | | | Destination | | | | | | | | | | | | | | | | | |
|-----|--------|----|----|----|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|
| | 15-04 | 03 | 02 | 01 | 00 | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | |
| | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | x | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | x | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | x | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | x | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | x | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | x | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Entering Parameters

- **Source** is the address that contains the information to be decoded. Only the first four bits (0–3) are used by the DCD instruction. The remaining bits may be used for other application specific needs.
- **Destination** is the address of the word where the decoded data is to be stored.

Updates to Arithmetic Status Bits

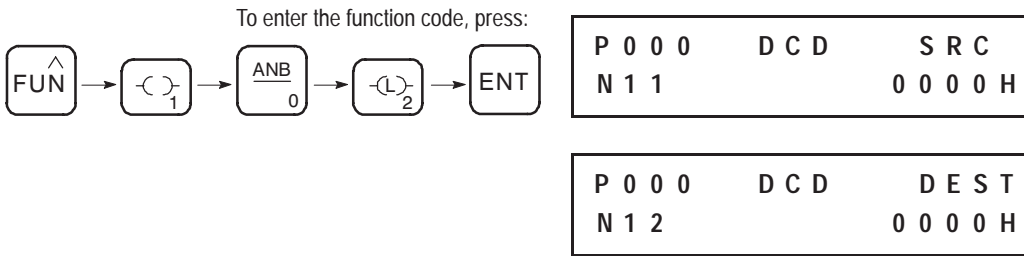
Unaffected.

Entering the Instruction

You enter the instruction from within the program monitor functional area. As you enter the instruction, you can return to previously entered operands by pressing this key:

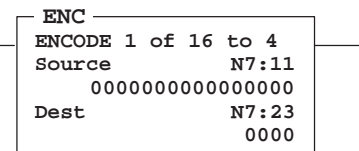


Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters you must go into the overwrite mode. (See page 17–4.)



Encode 1 of 16 to 4 (ENC)

Ladder representation:



Execution Times (μsec) when:

| True | False |
|-------|-------|
| 54.80 | 6.78 |

When the rung is true, this output instruction searches the source from the lowest to the highest bit and looks for the first set bit. The corresponding bit position is written to the destination as an integer, as shown in the table below.

| Bit | Source | | | | | | | | | | | | | | | | Destination | | | | | |
|-----|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|---|
| | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | 15-04 | 03 | 02 | 01 | 00 | |
| x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 1 | x | 0 | 0 | 0 | 0 | |
| x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 1 | 0 | x | 0 | 0 | 0 | 1 |
| x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 1 | 0 | 0 | x | 0 | 0 | 1 | 0 |
| x | x | x | x | x | x | x | x | x | x | x | x | x | 1 | 0 | 0 | 0 | 0 | x | 0 | 0 | 1 | 1 |
| x | x | x | x | x | x | x | x | x | x | x | 1 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | 1 | 0 | 1 |
| x | x | x | x | x | x | x | x | x | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | 1 | 1 | 0 |
| x | x | x | x | x | x | x | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | 1 | 1 | 1 |
| x | x | x | x | x | x | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 1 | 0 | 0 | 1 |
| x | x | x | x | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 1 | 0 | 1 | 0 |
| x | x | x | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 1 | 1 | 0 | 1 |
| x | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 1 | 1 | 1 | 1 |

Entering Parameters

- **Source** is the address of the word to be encoded. Only one bit of this word should be on at any one time. If more than one bit in the source is set, the destination bits are set based on the least significant bit that is set. If a source of zero is used, all of the destination bits are reset and the zero bit is set.
- **Destination** is the address that contains the bit encode information. Bits 4–15 of the destination are reset by the ENC instruction.

Updates to Arithmetic Status Bits

The arithmetic status bits are found in Word 0, bits 0–3 in the controller status file. After an instruction is executed, the arithmetic status bits in the status file are updated:

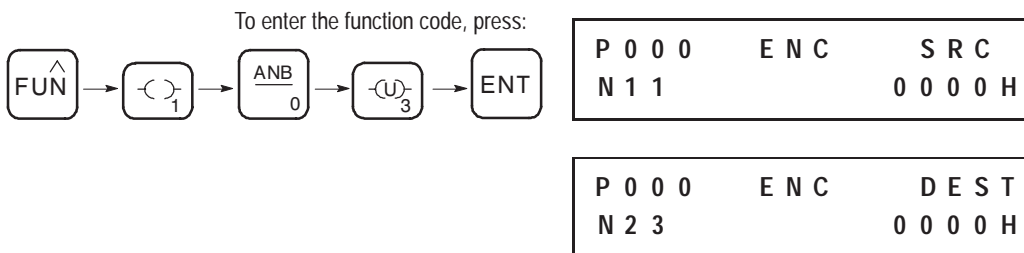
| With this Bit: | The Controller: |
|-------------------|--|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | sets if more than one bit in the source is set; otherwise reset. The math overflow bit (S5/0) is <i>not</i> set. |
| S0/2 Zero (Z) | sets if destination value is zero. |
| S0/3 Sign (S) | always resets. |

Entering the Instruction

You enter the instruction from within the program monitor functional area. As you enter the instruction, you can return to previously entered operands by pressing this key:

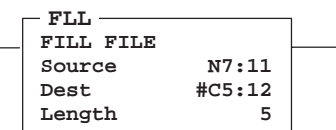
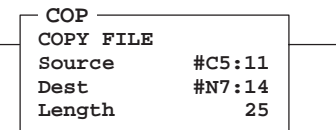


Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters you must go into the overwrite mode. (See page 17–4.)



Copy File (COP) and Fill File (FLL) Instructions

Ladder representation:



Execution Times (µsec) when:

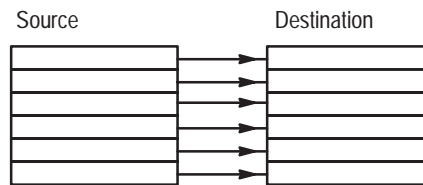
| | True | False |
|-----|-----------------|-------|
| COP | 27.31+5.06/word | 6.60 |
| FLL | 26.86+3.62/word | 6.60 |

The destination file type determines the number of words that an instruction transfers. For example, if the destination file type is a counter and the source file type is an integer, three integer words are transferred for each element in the counter-type file.

After a COP or FLL instruction is executed, index register S24 is cleared to zero.

Using COP

This instruction copies blocks of data from one location into another. It uses no status bits. If you need an enable bit, program an output instruction (OUT) in parallel using an internal bit as the output address. The following figure shows how file instruction data is manipulated.



File to File

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** is the address of the first word in the file to be copied. You must use the file indicator (#) in the address. (The HHP inserts the # character automatically.)
- **Destination** is the address of the first word in the file where the data is to be stored. You must use the file indicator (#) in the address. (The HHP inserts the # character automatically.)
- **Length** is the number of *words or elements* in the file to be copied. See the table below.

| If the destination file type is a(n): | then you can specify a maximum length of: | |
|---------------------------------------|---|--------|
| | Discrete | Analog |
| Output | 1 | 5 |
| Input | 2 | 8 |
| Status | 33 | — |
| Bit | 32 | — |
| Timer | 40 | — |
| Counter | 32 | — |
| Control | 16 | — |
| Integer | 105 | — |

Important: The maximum lengths apply when the source is of the same file type.

All elements are copied from the source file into the destination file each time the instruction is executed. Elements are copied in ascending order.

If your destination file type is a timer, counter, or control file, be sure that the destination words corresponding to the status elements of your source file contain zeros.

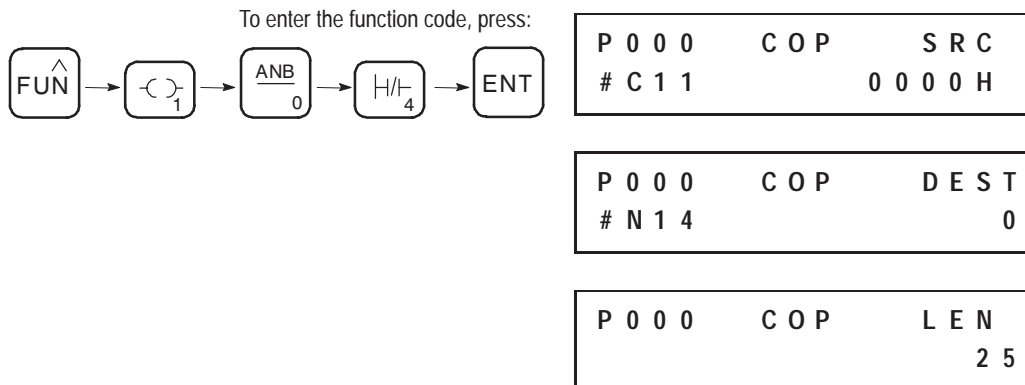
Entering the Instruction

You enter the instruction from within the program monitor functional area. The following items apply when entering the instruction:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:

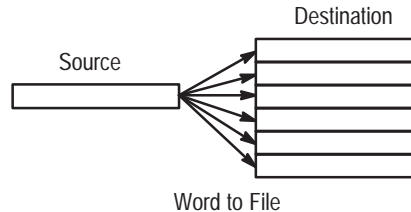


Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17-4.)



Using FLL

The following figure shows how file instruction data is manipulated. The instruction fills the words of a file with a source value. It uses no status bits. If you need an enable bit, program a parallel output that uses a storage address.



Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** is a constant or element address. The file indicator (#) is *not* required for an element address.
- **Destination** is the starting address of the file you want to fill. You must use the file indicator (#) in the address.
- **Length** is the number of *words or elements* in the file to be filled.

| If the destination file type is a: | then you can specify a maximum length of: | |
|------------------------------------|---|--------|
| | Discrete | Analog |
| Output | 1 | 5 |
| Input | 2 | 8 |
| Status | 33 | — |
| Bit | 32 | — |
| Timer | 40 | — |
| Counter | 32 | — |
| Control | 16 | — |
| Integer | 105 | — |

All elements are filled from the source value (typically a constant) into the specified destination file each scan the rung is true. Elements are filled in ascending order.

Entering the Instruction

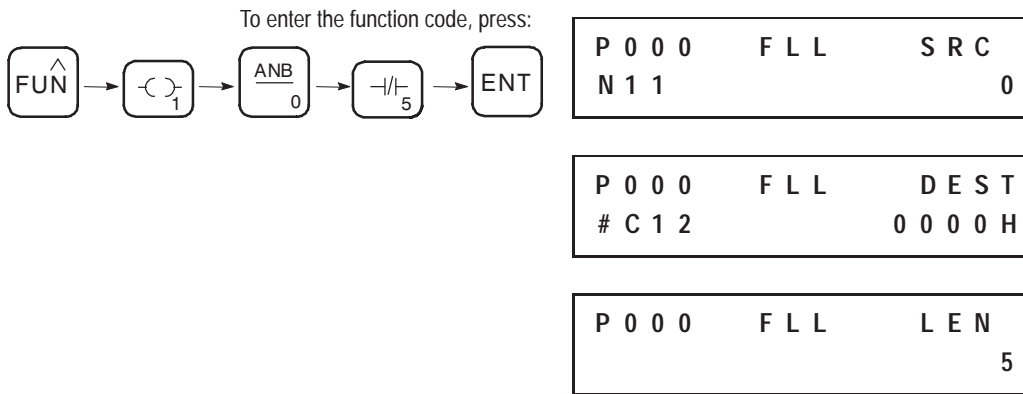
You enter the instruction from within the program monitor functional area. The following items apply when entering the instruction:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the

instruction's parameters, you must go into the overwrite mode. (See page 17-4.)



Move and Logical Instructions Overview

The following general information applies to move and logical instructions.

Entering Parameters

- **Source** is the address of the value on which the logical or move operation is to be performed. It can be a word address or a constant. If the instruction has two source operands, it will not accept constants in both operands.
- **Destination** is the address where the resulting data is stored. It must be a word address.

Entering the Instructions

You enter the instructions from within the program monitor functional area. The following items apply when entering the instructions:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17-4.)

Using Indexed Word Addresses

You have the option of using indexed word addresses for instruction parameters specifying word addresses. Indexed addressing is discussed in chapter 6.

Updates to Arithmetic Status Bits

The arithmetic status bits are found in Word 0, bits 0–3 in the controller status file. After an instruction is executed, the arithmetic status bits in the status file are updated:

| Bit | Name | Description |
|------|--------------|--|
| S0/0 | Carry (C) | Set if a carry is generated; otherwise cleared. |
| S0/1 | Overflow (V) | Indicates that the actual result of a math instruction does not fit in the designated destination. |
| S0/2 | Zero (Z) | Indicates a 0 value after a math, move, or logic instruction. |
| S0/3 | Sign (S) | Indicates a negative (less than 0) value after a math, move, or logic instruction. |

Overflow Trap Bit, S5/0

Minor error bit (S5/0) is set upon detection of a mathematical overflow or division by zero. If this bit is set upon execution of an END statement or a Temporary End (TND) instruction, the recoverable major error code 0020 is declared.

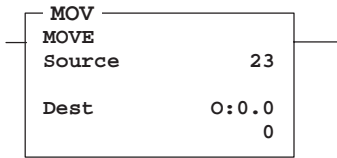
In applications where a math overflow or divide by zero occurs, you can avoid a controller fault by using a reset (RST) instruction with address S5/0 in your program. The rung must be between the overflow point and the END or TND statement.

Changes to the Math Register, S13 and S14

Move and logical instructions do not affect the math register.

Move (MOV)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 25.05 | 6.78 |

This output instruction moves the source data to the destination location. As long as the rung remains true, the instruction moves the data each scan.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** is the address or constant of the data you want to move.
- **Destination** is the address where the instruction moves the data.

Tip

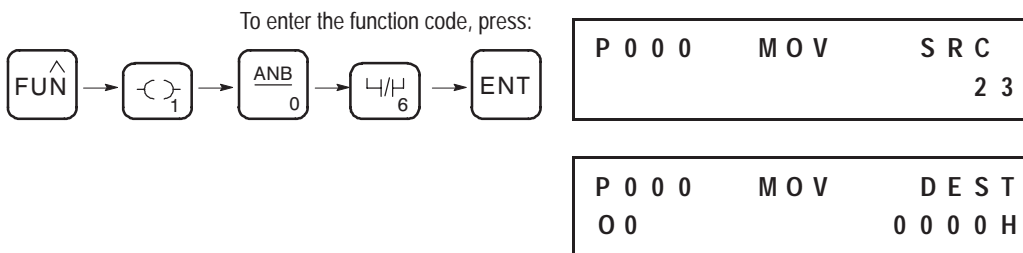
If you wish to move one word of data without affecting the math flags, use a copy (COP) instruction with a length of 1 word instead of the MOV instruction.

Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|---|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | always resets. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets. |
| S0/3 Sign (S) | sets if result is negative (most significant bit is set); otherwise resets. |

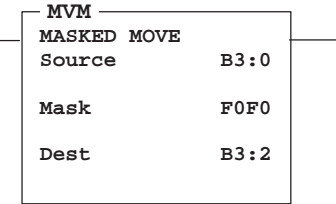
Entering the Instruction

You enter the instruction from within the program monitor functional area.



Masked Move (MVM)

Ladder representation:



Execution Times (µsec) when:

| | |
|-------|-------|
| True | False |
| 33.28 | 6.78 |

The MVM instruction is a word instruction that moves data from a source location to a destination and allows portions of the destination data to be masked by a separate word. As long as the rung remains true, the instruction moves the data each scan.

Entering Parameters

Enter the following parameters when programming this instruction:

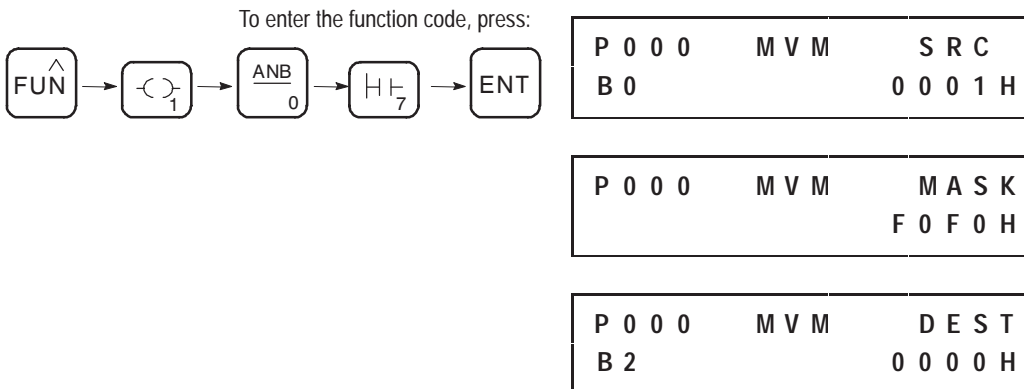
- **Source** is the address of the data you want to move.
- **Mask** is the address of the mask through which the instruction moves data; the mask can be a hex value (constant).
- **Destination** is the address where the instruction moves the data.

Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|---|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | always resets. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets. |
| S0/3 Sign (S) | sets if result is negative; otherwise resets. |

Entering the Instruction

You enter the instruction from within the program monitor functional area.



Operation

When the rung containing this instruction is true, data at the source address passes through the mask to the destination address. See the following HHP displays and figure.

| | | |
|---------|-------|-----------|
| P 0 0 0 | M V M | S R C |
| B 0 | | 5 5 5 5 H |

| | | |
|---------|-------|-----------|
| P 0 0 0 | M V M | M A S K |
| | | F 0 F 0 H |

| | | |
|---------|-------|-----------|
| P 0 0 0 | M V M | D E S T |
| B 2 | | F F F F H |

B2 before move

| |
|---------------------------------|
| 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |
|---------------------------------|

source B0

| |
|---------------------------------|
| 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 |
|---------------------------------|

Mask F0F0

| |
|---------------------------------|
| 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 |
|---------------------------------|

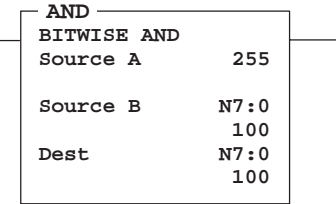
B2 after move

| |
|---------------------------------|
| 0 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 |
|---------------------------------|

Mask data by setting bits in the mask to zero; pass data by setting bits in the mask to one. The mask can be a constant value or you can vary the mask by assigning a direct address. *Bits in the destination that correspond to zeros in the mask are not altered.*

And (AND)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 34.00 | 6.78 |

The value at source A is ANDed bit by bit with the value at source B and then stored in the destination. (This instruction differs from the AND *input* instruction discussed in chapter 8.)

Truth Table

| Dest = A AND B | | |
|----------------|---|------|
| A | B | Dest |
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

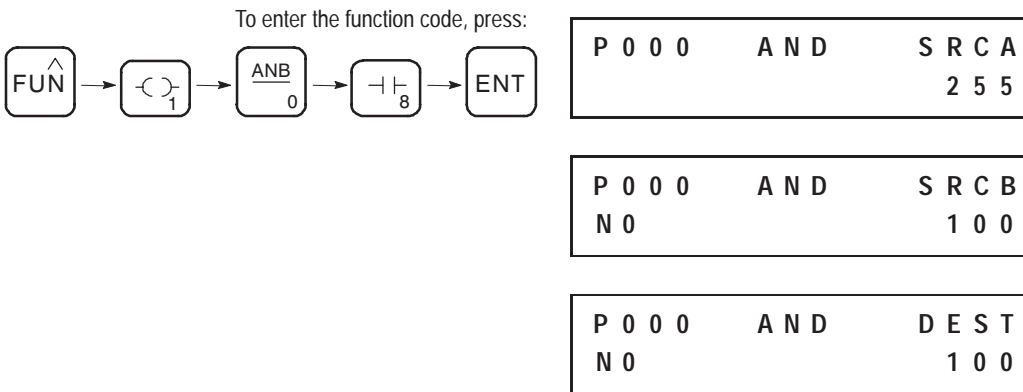
Source A and B can either be a word address or a constant; however, both sources cannot be a constant. The destination must be a word address.

Updates to Arithmetic Status Bits

| With this Bit: | The controller: |
|-------------------|--|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | always resets. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets. |
| S0/3 Sign (S) | sets if most significant bit is set; otherwise resets. |

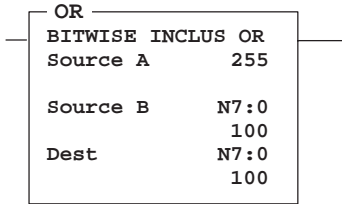
Entering the Instruction

You enter the instruction from within the program monitor functional area.



Or (OR)

Ladder representation:



Execution Times (µsec) when:

| | |
|-------|-------|
| True | False |
| 33.68 | 6.78 |

The value at source A is ORed bit by bit with the value at source B and then stored in the destination. (This instruction differs from the OR *input* instruction discussed in chapter 8.)

Truth Table

| Dest = A OR B | | |
|---------------|---|------|
| A | B | Dest |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

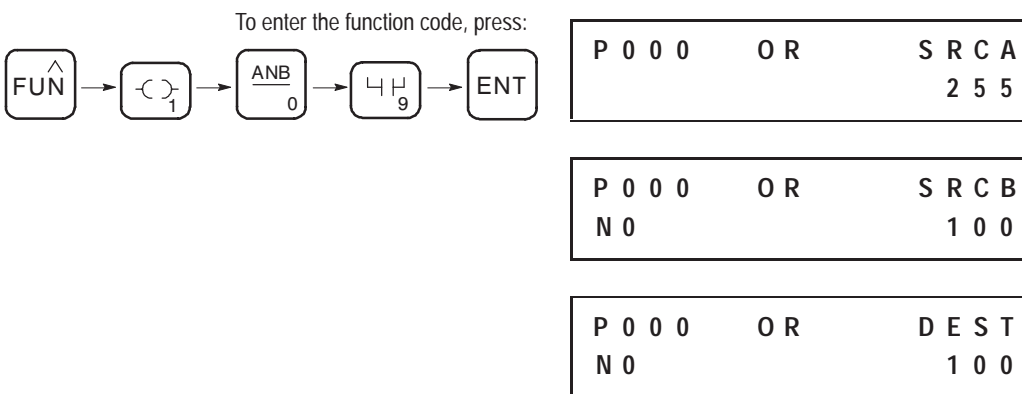
Source A and B can either be a word address or a constant; however, both sources cannot be a constant. The destination must be a word address.

Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|--|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | always resets. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets. |
| S0/3 Sign (S) | sets if result is negative (most significant bit is set) otherwise resets. |

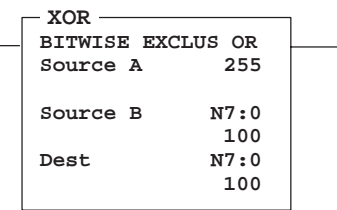
Entering the Instruction

You enter the instruction from within the program monitor functional area.



Exclusive Or (XOR)

Ladder representation:



Execution Times (μsec) when:

| | |
|-------|-------|
| True | False |
| 33.64 | 6.92 |

The value at source A is Exclusive ORed bit by bit with the value at source B and then stored in the destination.

Truth Table

| Dest = A XOR B | | |
|----------------|---|------|
| A | B | Dest |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

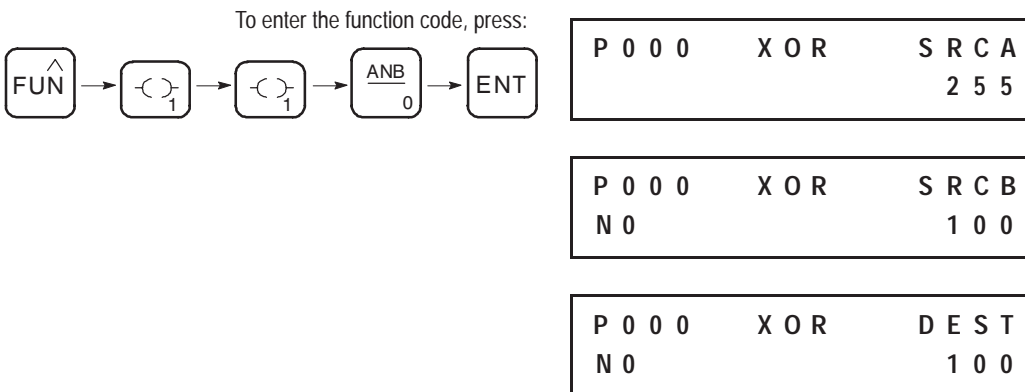
Source A and B can either be a word address or a constant; however, both sources cannot be a constant. The destination must be a word address.

Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|---|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | always resets. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets |
| S0/3 Sign (S) | sets if result is negative (most significant bit is set); otherwise resets. |

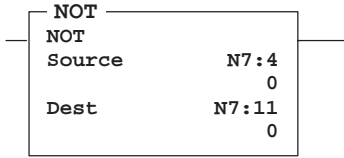
Entering the Instruction

You enter the instruction from within the program monitor functional area.



Not (NOT)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 28.21 | 6.78 |

The source value is NOTed bit by bit and then stored in the destination (one's complement).

Truth Table

| Dest = NOT A | |
|--------------|------|
| A | Dest |
| 0 | 1 |
| 1 | 0 |

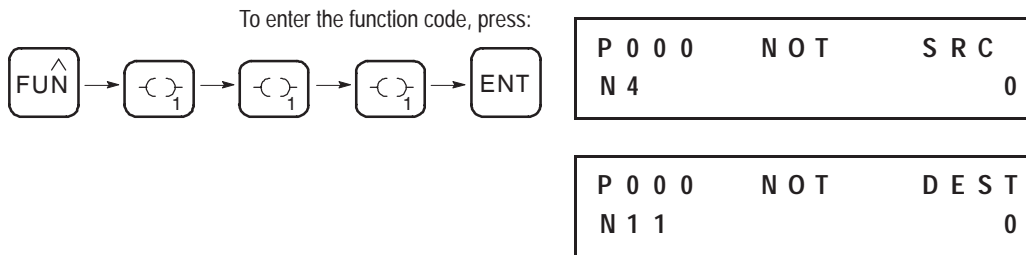
The source and destination must be word addresses.

Updates to Arithmetic Status Bits

| With this Bit: | The Controller: |
|-------------------|---|
| S0/0 Carry (C) | always resets. |
| S0/1 Overflow (V) | always resets. |
| S0/2 Zero (Z) | sets if result is zero; otherwise resets. |
| S0/3 Sign (S) | sets if result is negative (most significant bit is set); otherwise resets. |

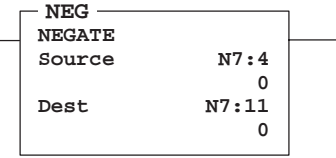
Entering the Instruction

You enter the instruction from within the program monitor functional area.



Negate (NEG)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 29.48 | 6.78 |

Use the NEG instruction to change the sign of a value. If you negate a negative value, the result is a positive; if you negate a positive value, the result is a negative. The destination contains the two's complement of the source.

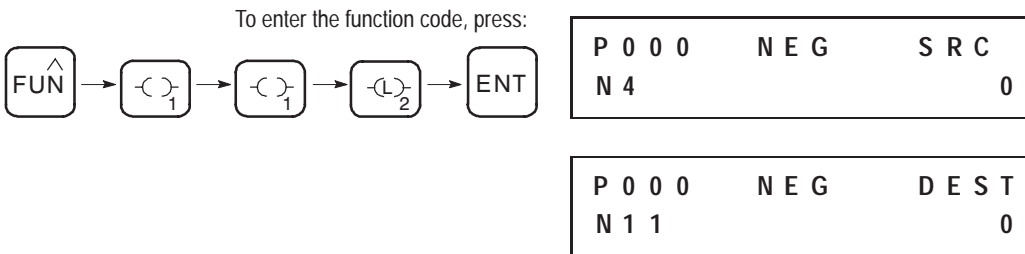
The source and destination must be word addresses.

Updates to Arithmetic Status Bits

| With this Bit: | | The Controller: |
|----------------|--------------|--|
| S0/0 | Carry (C) | clears if 0 or overflow, otherwise sets. |
| S0/1 | Overflow (V) | sets if overflow, otherwise reset. Overflow occurs only if -32,768 is the source. On overflow, the minor error flag is also set. The value 32,767 is placed in the destination. If S14 is set, then the unsigned, truncated overflow remains in the destination. |
| S0/2 | Zero (Z) | sets if result is zero; otherwise resets. |
| S0/3 | Sign (S) | sets if result is negative; otherwise resets. |

Entering the Instruction

You enter the instruction from within the program monitor functional area.



FIFO and LIFO Instructions Overview

FIFO instructions load words into a file and unload them in the same order as they were loaded. The first word in is the first word out.

LIFO instructions load words into a file and unload them in the opposite order as they were loaded. The last word in is the first word out.

Entering Parameters

Enter the following parameters when programming these instructions:

- **Source** is a word address or constant (–32,768 to 32,767) that becomes the next value in the stack.
- **Destination (Dest)** is a word address that stores the value that exits from the stack.

| This Instruction: | Unloads the Value from: |
|-------------------|-------------------------|
| FIFO's FFU | First word |
| LIFO's LFU | The last word entered |

- **FIFO/LIFO** is the address of the stack. It must be an indexed word address in the bit, input, output, or integer file. (The HHP inserts the # character automatically.) Use the same FIFO address for the associated FFL and FFU instructions; use the same LIFO address for the associated LFL and LFU instructions.
- **Control** is the address of the control structure. The control structure stores the status bits, the stack length, and the position value. Do not use the control file address for any other instruction.
Status bits of the control structure are addressed by mnemonic. These include:
 - **Empty Bit EM** (bit 12) is set by the controller to indicate the stack is empty.
 - **Done Bit DN** (bit 13) is set by the controller to indicate the stack is full. This inhibits loading the stack.
 - **FFU/LFU Enable Bit EU** (bit 14) is set on a false-to-true transition of the FFU/LFU rung and is reset on a true-to-false transition.
 - **FFL/LFL Enable Bit EN** (bit 15) is set on a false-to-true transition of the FFL/LFL rung and is reset on a true-to-false transition.
- **Length** specifies the maximum number of words in the stack. Address the length value by mnemonic (LEN).
- **Position** is the next available location where the instruction loads data into the stack. This value changes after each load or unload operation. Address the position value by mnemonic (POS).

Entering the Instructions

The following items apply when entering the instructions:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17–4.)

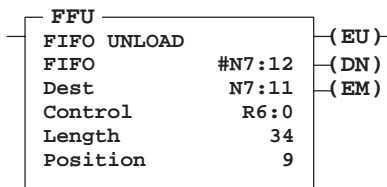
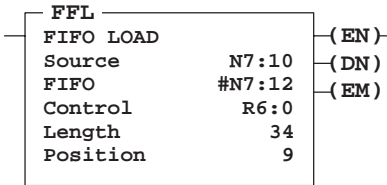
Effects on Index Register S24

The value present in S24 is overwritten with the position value when a false-to-true transition of the FFL/FFU or LFL/LFU rung occurs. For the FFL/LFL, the position value determined at instruction entry is placed in S24. For the FFU/LFU, the position value determined at instruction exit is placed in S24.

When the DN bit is set, a false-to-true transition of the FFL/LFL rung does not change the position value or the index register value. When the EM bit is set, a false-to-true transition of the FFU/LFU rung does not change the position value or the index register value.

FIFO Load (FFL) and FIFO Unload (FFU)

Ladder representation:



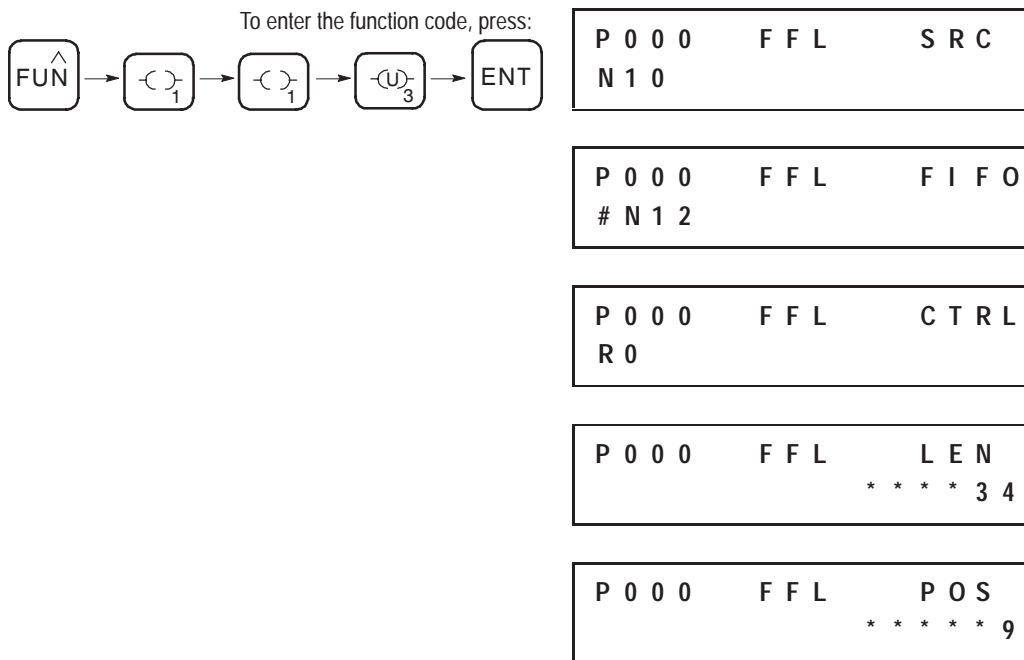
Execution Times (µsec) when:

| | True | False |
|-----|--------------------------------|-------|
| FFL | 61.13 | 33.67 |
| FFU | 73.78+4.34 x position value | 34.90 |

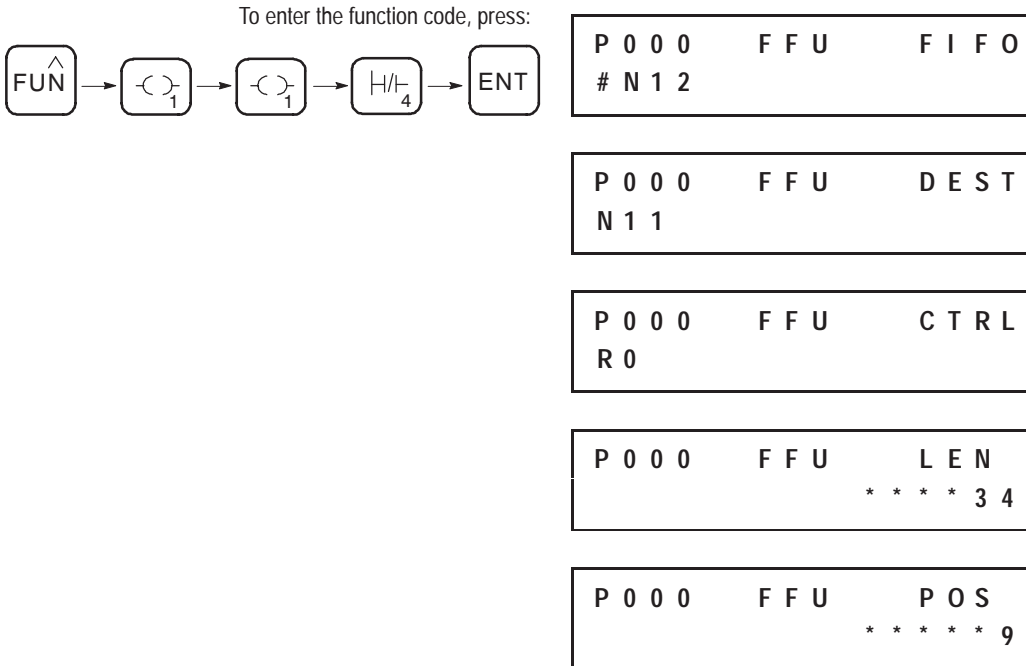
FFL and FFU instructions are used in pairs. The FFL instruction loads words into a user-created file called a FIFO stack. The FFU instruction unloads words from the FIFO stack in the same order as they were entered.

Entering the Instructions

You enter the instruction from within the program monitor functional area. While entering the FFL instruction, you see these screens:

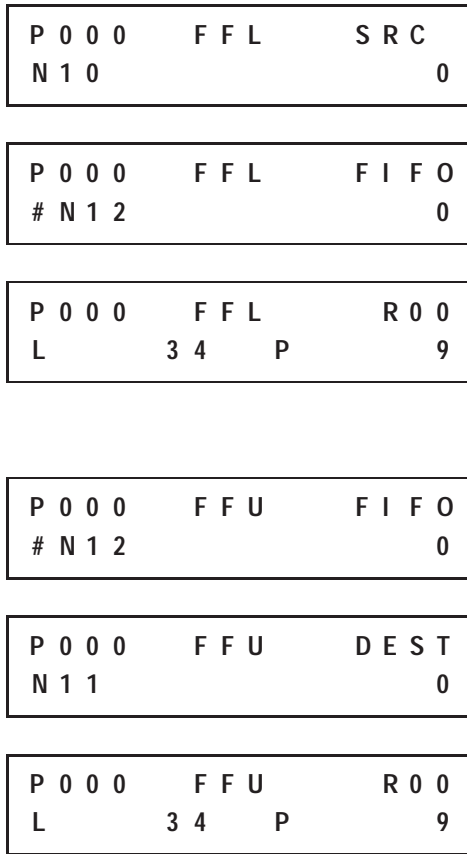


While entering the FFU instruction, you see these screens:

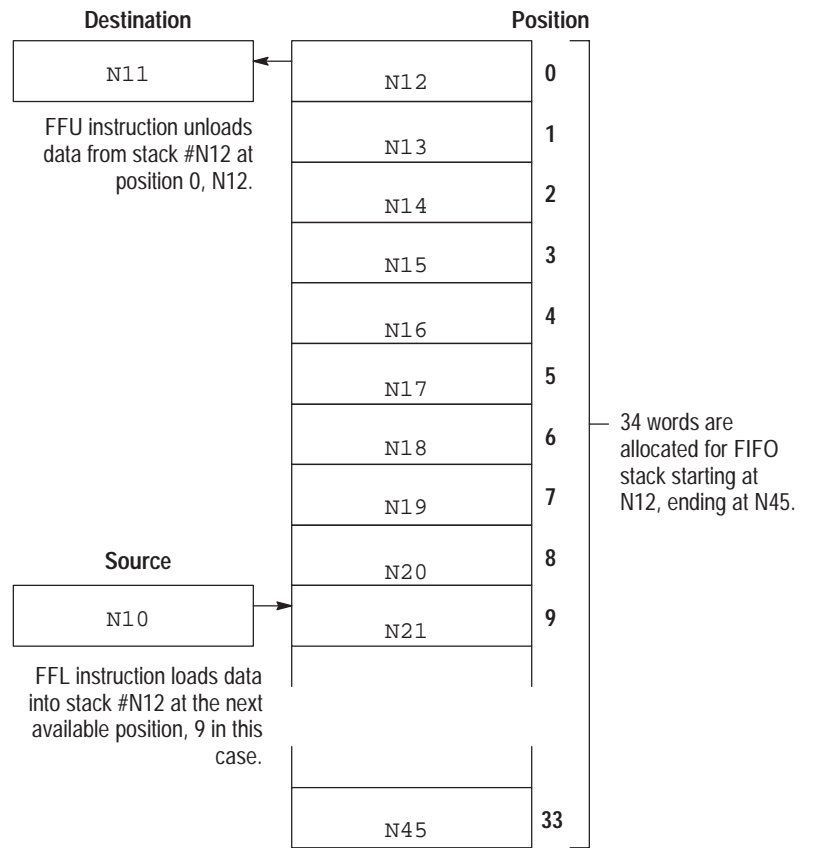


Operation

The operation of the FFL – FFU instruction pair is shown on the following page. The screens shown to the left of the figure are the condensed screens that appear after instruction entry is complete.



FFL–FFU Instruction Pair



Loading and Unloading of Stack #N12

FFL Instruction

When rung conditions change from false-to-true, the controller sets the FFL enable bit (EN). This loads the contents of the Source, N10, into the stack structure indicated by the position number, 9. The position value then increments.

The FFL instruction loads an element at each false-to-true transition of the rung, until the stack is filled (34 elements). The controller then sets the done bit (DN), inhibiting further loading.

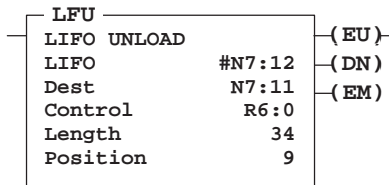
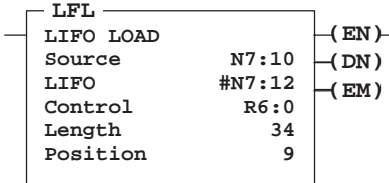
FFU Instruction

When rung conditions change from false-to-true, the controller sets the FFU enable bit (EU). This unloads the contents of the element at stack position 0 into the Destination, N11. All data in the stack is shifted one element toward position zero and the highest numbered element is zeroed. The position value then decrements.

The FFU instruction unloads an element at each false-to-true transition of the rung, until the stack is empty. The controller then sets the empty bit (EM).

LIFO Load (LFL) and LIFO Unload (LFU)

Ladder representation:



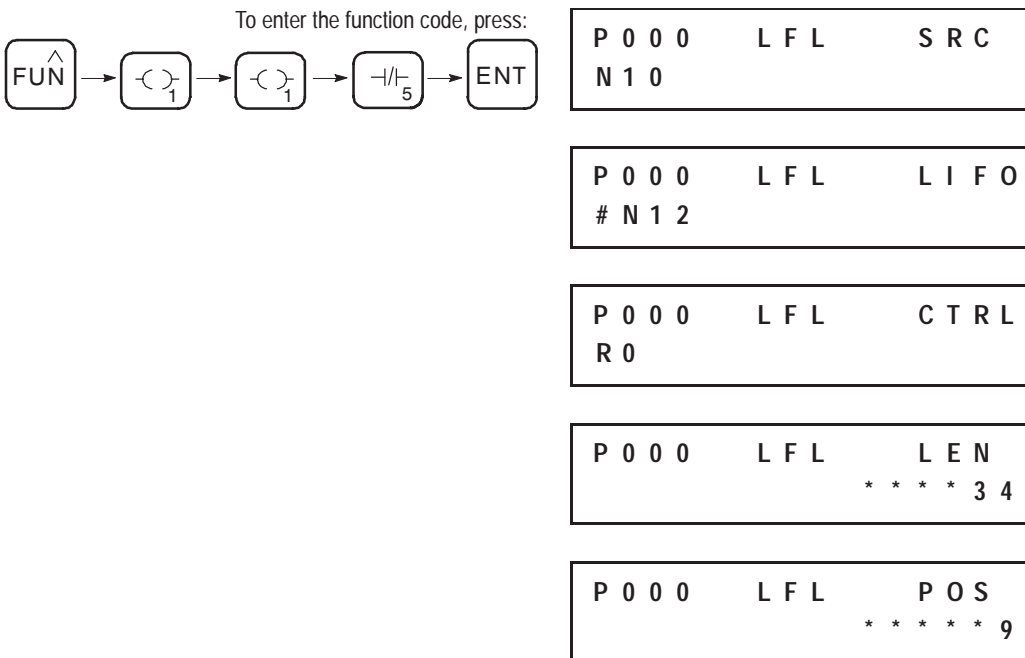
Execution Times (µsec) when:

| | True | False |
|-----|-------|-------|
| LFL | 61.13 | 33.67 |
| LFU | 64.20 | 35.08 |

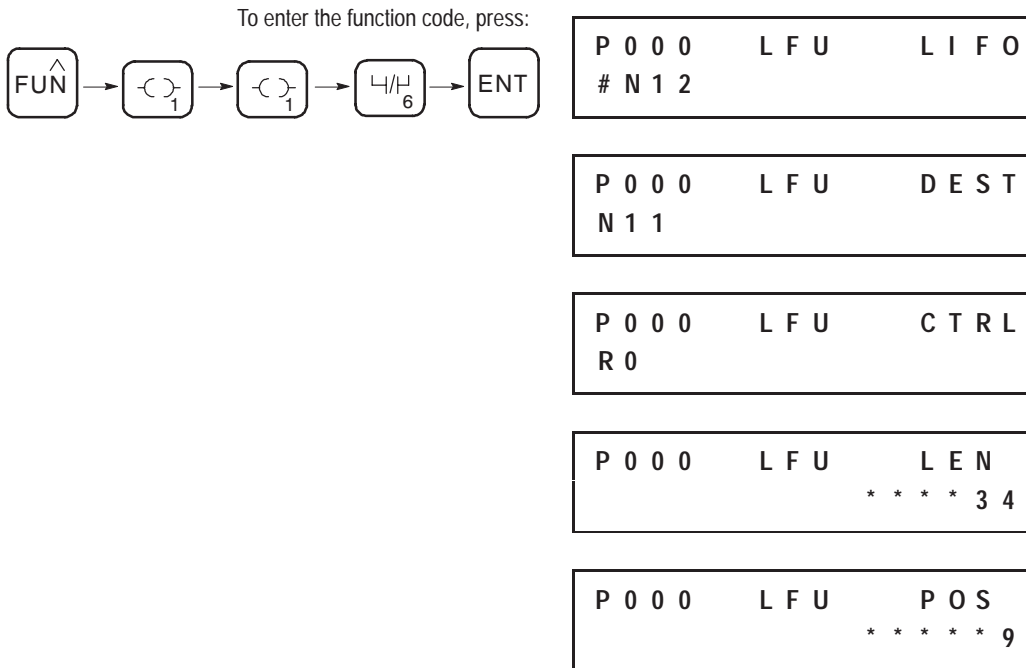
LFL and LFU instructions are used in pairs. The LFL instruction loads words into a user-created file called a LIFO stack. The LFU instruction unloads words from the LIFO stack in the opposite order as they were entered.

Entering the Instructions

You enter the instructions from within the program monitor functional area. While entering the LFL instruction, you see these screens:

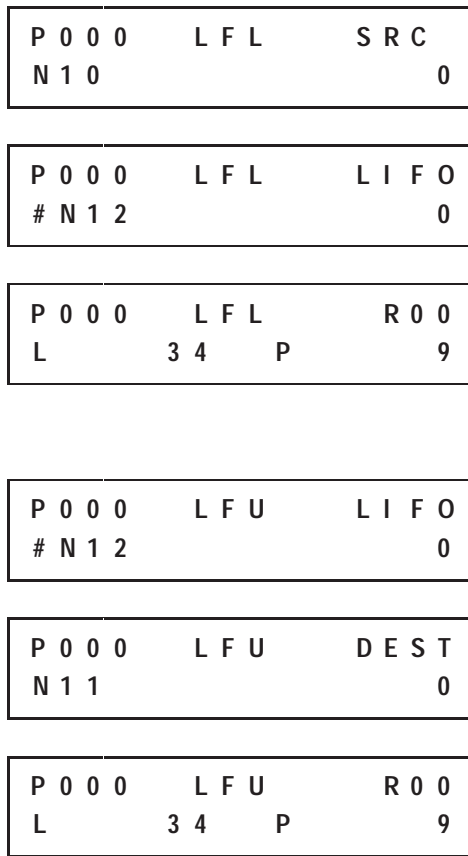


While entering the LFU instruction, you see these screens:

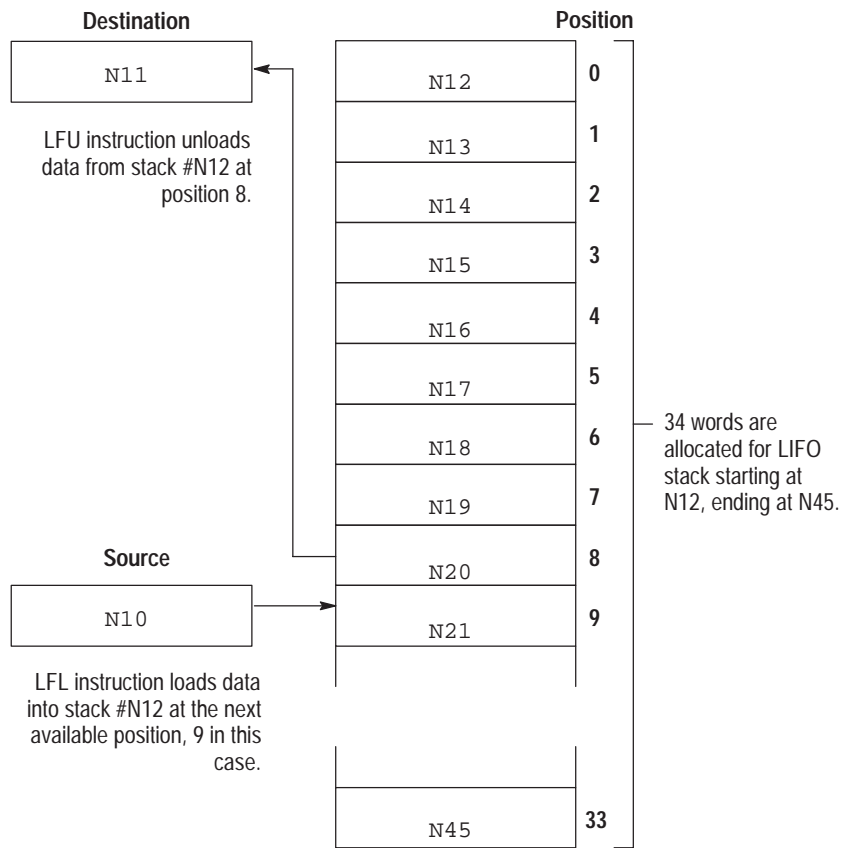


Operation

The operation of the LFL – LFU instruction pair is shown on the following page. The screens shown to the left of the figure are the condensed screens that appear after instruction entry is complete.



LFL-LFU Instruction Pair



Loading and Unloading of Stack #N12

LFL Instruction

When rung conditions change from false-to-true, the controller sets the LFL enable bit (EN). This loads the contents of the Source, N10, into the stack element indicated by the position number, 9. The position value then increments.

The LFL instruction loads an element at each false-to-true transition of the rung, until the stack is filled (34 elements). The controller sets the done bit (DN), inhibiting further loading.

LFU Instruction

When rung conditions change from false-to-true, the controller sets the LFU enable bit (EU). This unloads data from the last element loaded into the stack (at the position value minus 1), placing it in the Destination, N11. The position value then decrements.

The LFU instruction unloads one element at each false-to-true transition of the rung, until the stack is empty. The controller then sets the empty bit (EM).

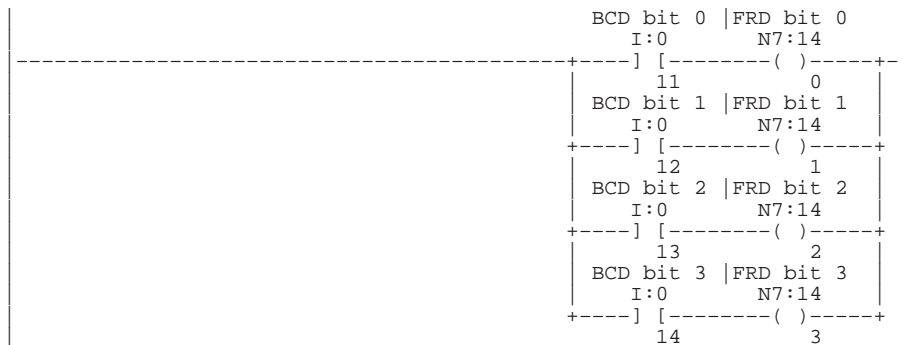
Data Handling Instructions in the Paper Drilling Machine Application Example

To demonstrate the use of data handling instructions, this section provides ladder rungs, followed by the optimized instruction list for these rungs. The rungs are part of the paper drilling machine application example described in appendix E. You will be adding to the subroutine in file 7 that was started in chapter 9.

Ladder Rungs

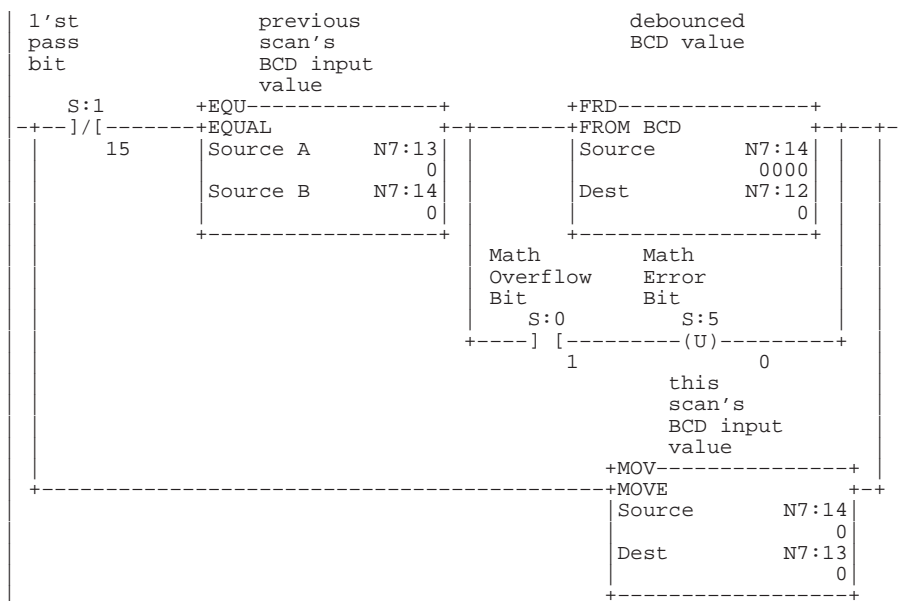
Rung 7:2^①

Moves the single-digit BCD thumbwheel value into an internal integer register. This is done to properly align the four BCD input signals prior to executing the BCD to Integer instruction (FRD). The thumbwheel is used to allow the operator to enter the thickness of the paper that is to be drilled. The thickness is entered in 1/4 in. increments. This provides a range of 1/4 in. to 2.25 in.



Rung 7:3

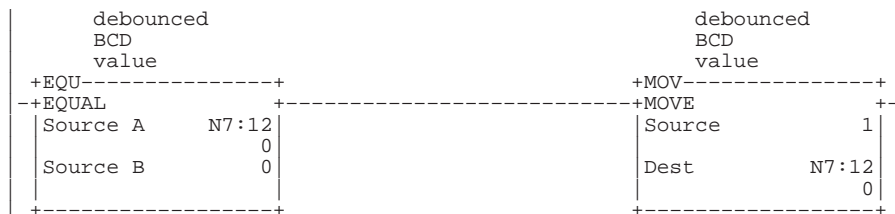
Converts the BCD thumbwheel value from BCD to integer. This is done because the controller operates upon integer values. This rung also "debounces" the thumbwheel to ensure that the conversion only occurs on valid BCD values. Note that invalid BCD values can occur while the operator is changing the BCD thumbwheel. This is due to input filter propagation delay differences between the four input circuits that provide the BCD input value.



^① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.

Rung 7:4

Ensures that the operator cannot select a paper thickness of 0. If this were allowed, the drill bit life calculation could be defeated resulting in poor quality holes due to a dull drill bit. Therefore the minimum paper thickness used to calculate drill bit wear is 1/4 in.



Instruction List

File 7, Rung 2^①

Moves the single digit BCD thumbwheel value into an internal integer register. This is done to properly align the four BCD input signals prior to executing the BCD to Integer instruction (FRD). The thumbwheel is used to allow the operator to enter the thickness of the paper that is to be drilled. The thickness is entered in 1/4 in. increments. This provides a range of 1/4 in. to 2.25 in.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| 10 | | MPS | | | |
| 22 | -] [- | AND | BCD bit 0 I/11 | 0 | |
| 40 | -()- | OUT | FRD bit 0 N14/0 | 0 | |
| 11 | | MRD | | | |
| 22 | -] [- | AND | BCD bit 1 I/12 | 0 | |
| 40 | -()- | OUT | FRD bit 1 N14/1 | 0 | |
| 11 | | MRD | | | |
| 22 | -] [- | AND | BCD bit 2 I/13 | 0 | |
| 40 | -()- | OUT | FRD bit 2 N14/2 | 0 | |
| 12 | | MPP | | | |
| 22 | -] [- | AND | BCD bit 3 I/14 | 0 | |
| 40 | -()- | OUT | FRD bit 3 N14/3 | 0 | |

① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.

File 7, Rung 3

Converts the BCD thumbwheel value from BCD to integer. This is done because the controller operates upon integer values. This rung also "debounces" the thumbwheel to ensure that the conversion only occurs on valid BCD values. Note that invalid BCD values can occur while the operator is changing the BCD thumbwheel. This is due to input filter propagation delay differences between the four input circuits that provide the BCD input value.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|----------------|--------|
| 10 | | MPS | | | |
| 23 | -]/[- | ANI | 1'st pass bit S1/15 | 0 | |
| 51 | -EQU- | AND-EQU | previous scan's BCD input value SRCA N13 SRCB N14 | 0000H 0000H | |
| 101 | | FRD | debounced BCD value SRC N14 DEST N12 | 0000H 0000H | |
| 22 | -] [- | AND | Math Overflow Bit S0/1 | 0 | |
| 42 | -(U)- | RST | Math Error Bit S5/0 | 0 | |
| 12 | | MPP | | | |
| 106 | | MOV | this scan's BCD input value SRC N14 DEST N13 | 0000H 0000H | |

File 7, Rung 4

Ensures that the operator cannot select a paper thickness of 0. If this were allowed, the drill bit life calculation could be defeated, resulting in poor quality holes due to a dull drill bit. Therefore, the minimum paper thickness used to calculate drill bit wear is 1/4 in.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|------------|--------|
| 50 | -EQU- | LD-EQU | debounced BCD value SRCA N12 SRCB | 0000H 0 | |
| 106 | | MOV | debounced BCD value SRC DEST N12 | 1 0000H | |

Using Program Flow Control Instructions

This chapter contains general information about the program flow instructions and explains how they function in your application program. Each of the instructions includes information on:

- what the instruction symbol looks like
- typical execution time for the instruction
- how to use the instruction
- how to enter the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the program flow control instructions in use.

Program Flow Control Instructions

| HPH Display | Mnemonic | Function Code | Name | Purpose | Page |
|-------------|---------------|---------------|----------------------------|--|------|
| ┌LBL┐ | JMP | 130 | Jump to Label | Jump forward or backward to the specified label instruction. | 12-2 |
| | LD LBL | 131 | Label | | |
| ┌SBR┐ | JSR | 132 | Jump to Subroutine | Jump to a designated subroutine and return. | 12-3 |
| | LD SBR | 133 | Subroutine | | |
| | RET | 134 | Return from Subroutine | | |
| | MCR | 135 | Master Control Reset | Turn off all non-retentive outputs in a section of ladder program. | 12-6 |
| | TND | 136 | Temporary End | Mark a temporary end that halts program execution. | 12-7 |
| | SUS | 137 | Suspend | Identifies specific conditions for program debugging and system troubleshooting. | 12-7 |
| | IIM | 138 | Immediate Input with Mask | Program an Immediate Input with Mask. | 12-8 |
| | IOM | 139 | Immediate Output with Mask | Program an Immediate Output with Mask. | 12-9 |

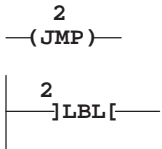
About the Program Flow Control Instructions

Use these instructions to control the sequence in which your program is executed.

Since these are *output* instructions (except LD LBL and LD SBR), they do not have LD, AND, and OR equivalents.

Jump (JMP) and Label (LBL)

Ladder representation:



Execution Times (μsec) when:

| | True | False |
|-----|------|-------|
| JMP | 9.04 | 6.78 |
| LBL | 1.45 | 0.99 |

Use these instructions in pairs to skip portions of the ladder program.

| If the Rung Containing the Jump Instruction is: | Then the Program: |
|---|--|
| True | Skips from the rung containing the JMP instruction to the rung containing the designated LBL instruction and then continues executing. You can jump forward or backward. |
| False | Does not execute the JMP instruction. |

Jumping forward to a label saves program scan time by omitting a program segment until needed. Jumping backward lets the controller execute program segments repeatedly.

Important: Be careful not to jump backwards an excessive number of times. The watchdog timer could time out and fault the controller. Use a counter, timer, or the “program scan” register (system status register, word S3, bits 0–7) to limit the amount of time you spend looping inside of JMP/LBL instructions.

Entering Parameters

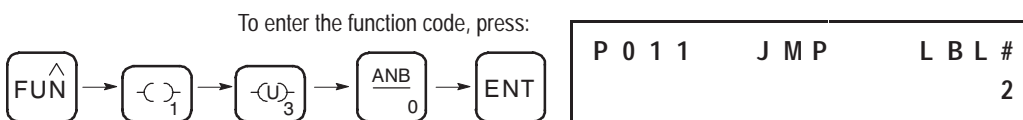
Enter a decimal label number from 0 to 999. You can place up to 1,000 labels in each subroutine file.

Using JMP

The JMP instruction causes the controller to skip rungs. You can jump to the same label from one or more JMP instruction.

Entering the Instruction

You enter the instruction from within the program monitor functional area. Asterisks appear on the display to indicate that the HHP is waiting for data entry (i.e., a number).



Using LBL

This input instruction is the target of JMP instructions having the same label number. This instruction has no control bits.

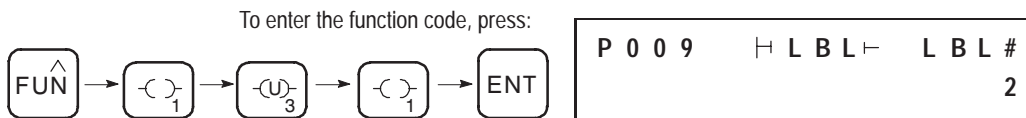
Important: You must program this instruction as the first instruction of a rung. (Since it must be the first instruction on the rung, the LBL instruction is also known as LD LBL.)

You can program multiple jumps to the same label by assigning the same label number to multiple JMP instructions. However, label numbers must be unique.

Important: Do not jump (JMP) into an MCR zone. Instructions that are programmed within the MCR zone starting at the LBL instruction and ending at the 'END MCR' instruction are always evaluated as though the MCR zone is true, regardless of the true state of the "Start MCR" instruction.

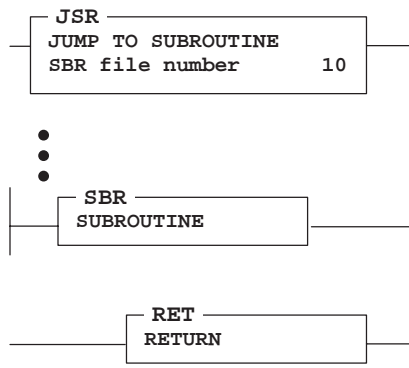
Entering the Instruction

You enter the instruction from within the program monitor functional area. Asterisks appear on the display to indicate that the HHP is waiting for data entry (i.e., a number).



Jump to Subroutine (JSR), Subroutine (SBR), and Return (RET)

Ladder representation:



The JSR, SBR, and RET instructions are used to direct the controller to execute a separate subroutine file within the ladder program and return to the instruction following the JSR instruction.

Important: If you use the SBR instruction, the SBR instruction must be the first instruction on the first rung in the program file that contains the subroutine. (Since it must be the first instruction on the rung, the SBR instruction is also known as LD SBR.)

Use a subroutine to store recurring sections of program logic that must be executed from several points within your application program. A subroutine saves memory because you program it only once.

Update critical I/O within subroutines using immediate input and/or output instructions (IIM, IOM), especially if your application calls for nested or relatively long subroutines. Otherwise, the controller does not update I/O until it reaches the end of the main program (after executing all subroutines).

Execution Times (μ sec) when:

| | True | False |
|-----|-------|-------|
| JSR | 22.24 | 4.25 |
| SBR | 1.45 | 0.99 |
| RET | 31.11 | 3.16 |



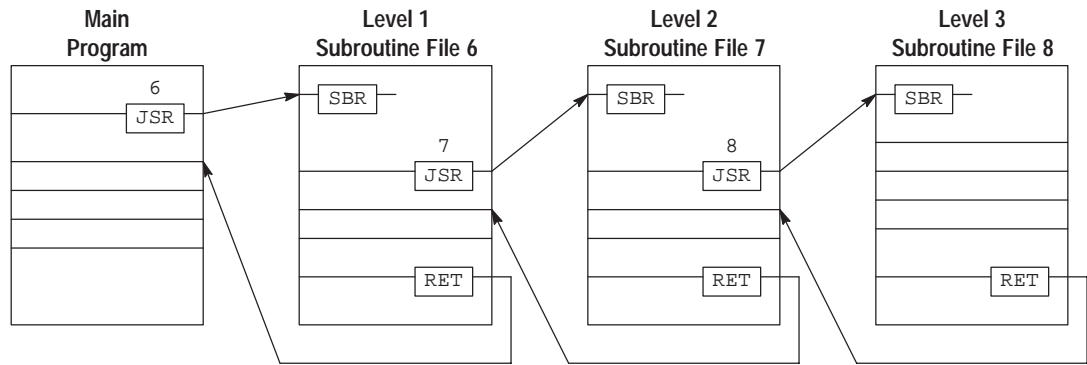
ATTENTION: Outputs controlled within a subroutine remain in their last state until the subroutine is executed again.

Nesting Subroutine Files

Nesting subroutines allows you to direct program flow from the main program to a subroutine and then on to another subroutine.

You can nest up to eight levels of subroutines. If you are using an STI subroutine, HSC interrupt subroutine, or user fault routine, you can nest subroutines up to three levels from each subroutine.

The following figure illustrates how subroutines may be nested.



Example of Nesting Subroutines to Level 3

An error occurs if more than the allowable levels of subroutines are called (subroutine stack overflow) or if more returns are executed than there are call levels (subroutine stack underflow).

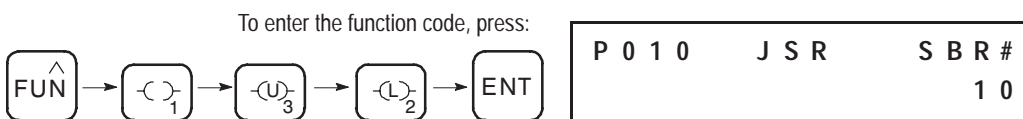
Using JSR

When the JSR instruction is executed, the controller jumps to the subroutine instruction (SBR) at the beginning of the target subroutine file and resumes execution at that point. You cannot jump into any part of a subroutine except the first instruction in that file.

You must program each subroutine in its own program file by assigning a unique file number (4–15).

Entering the Instruction

You enter the instruction from within the program monitor functional area. Asterisks appear on the display to indicate that the HHP is waiting for data entry (i.e., a number).



Using SBR

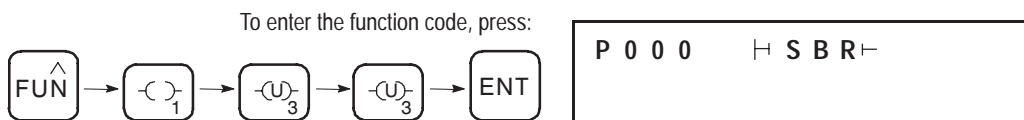
The target subroutine is identified by the file number that you entered in the JSR instruction. This instruction serves as a label or identifier for a program file as a regular subroutine file.

This instruction has no control bits. It is always evaluated as true. Use of this instruction is optional; however, we recommend using it for clarity.

Important: The instruction must be programmed as the first instruction of the first rung of a subroutine. (Since it must be the first instruction on the rung, the SBR instruction is also known as LD SBR.)

Entering the Instruction

You enter the instruction from within the program monitor functional area.



Using RET

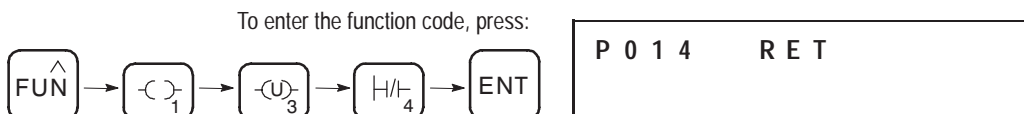
This output instruction marks the end of subroutine execution or the end of the subroutine file. It causes the controller to resume execution at the instruction following the JSR instruction.

The rung containing the RET instruction may be conditional if this rung precedes the end of the subroutine. In this way, the controller omits the balance of a subroutine only if its rung condition is true.

Without an RET instruction, the END instruction (always present in the subroutine) automatically returns program execution to the instruction following the JSR instruction in your calling ladder file.

Entering the Instruction

You enter the instruction from within the program monitor functional area.



Master Control Reset (MCR)

Ladder representation:

—(MCR)—

Execution Times (µsec) when:

| True | False |
|------|-------|
| 3.98 | 4.07 |

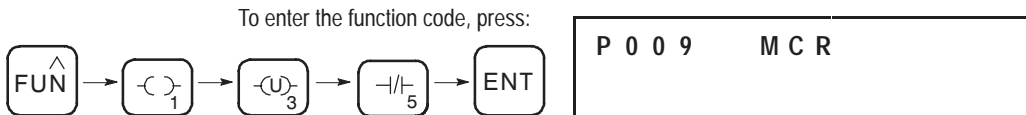
Use MCR instructions in pairs to create program zones that turn off all the non-retentive outputs in the zone. Rungs within the MCR zone are still scanned, but scan time is reduced due to the false state of non-retentive outputs. Non-retentive outputs are reset when their rung goes false.

| If the MCR Rung that Starts the Zone is: | Then the Controller: |
|--|---|
| True | Executes the rungs in the MCR zone based on each rung's individual input condition (as if the zone did not exist). |
| False | Resets all non-retentive output instructions in the MCR zone regardless of each rung's individual input conditions. |

MCR zones let you enable or inhibit segments of your program, such as for recipe applications.

Entering the Instruction

You enter the instruction from within the program monitor functional area.



When you program MCR instructions, note that:

- You must end the zone with an unconditional MCR instruction.
- You cannot nest one MCR zone within another.
- Do not jump into an MCR zone. If the zone is false, jumping into it activates the zone.

Important: The MCR instruction is not a substitute for a hard-wired master control relay that provides emergency stop capability. You still must install a hard-wired master control relay to provide emergency I/O power shutdown.



ATTENTION: If you start instructions such as timers or counters in an MCR zone, instruction operation ceases when the zone is disabled. Re-program critical operations outside the zone if necessary.

Temporary End (TND)

Ladder representation:

—(TND)—

Execution Times (μsec) when:

| True | False |
|------|-------|
| 7.78 | 3.16 |

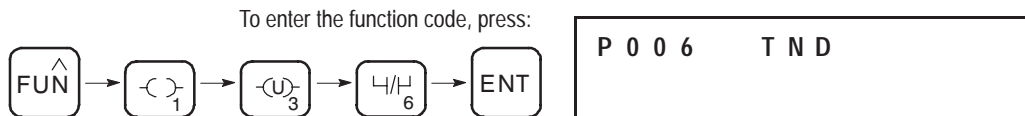
This instruction, when its rung is true, stops the controller from scanning the rest of the program file, updates the I/O, and resumes scanning at rung 0 of the main program (file 2). If this instruction's rung is false, the controller continues the scan until the next TND instruction or the END statement. Use this instruction to progressively debug a program or conditionally omit the balance of your current program file or subroutines.

Important: If you use this instruction inside a nested subroutine, execution of all nested subroutines is terminated.

Do not execute this instruction from the user error fault routine (file 3), high-speed counter interrupt routine (file 4), or selectable timed interrupt routine (file 5), or a fault will occur.

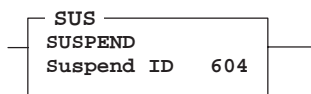
Entering the Instruction

You enter the instruction from within the program monitor functional area.



Suspend (SUS)

Ladder representation:



Execution Times (μsec) when:

| True | False |
|-------|-------|
| 10.85 | 7.87 |

When this instruction is executed, it causes the controller to enter the Suspend Idle mode and stores the Suspend ID in word 7 (S7) at the status file. All outputs are de-energized.

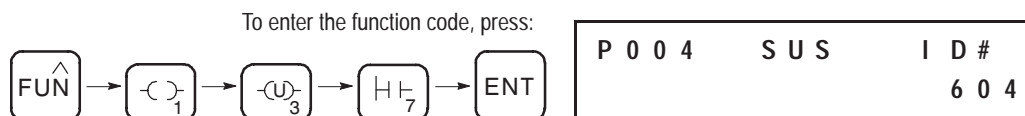
Use this instruction to trap and identify specific conditions for program debugging and system troubleshooting.

Entering Parameters

Enter a suspend ID number from -32,768 to +32,767 when you program the instruction.

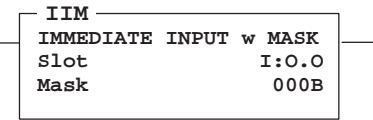
Entering the Instruction

You enter the instruction from within the program monitor functional area. Asterisks appear on the display to indicate that the HHP is waiting for data entry (i.e., a number).



Immediate Input with Mask (IIM)

Ladder representation:



Execution Times (μsec) when:

| True | False |
|-------|-------|
| 35.72 | 6.78 |

This instruction allows you to update data prior to the normal input scan. Data from a specified input is transferred through a mask to the input data file, making the data available to instructions following the IIM instruction in the ladder program.

For the mask, a 1 in an input's bit position passes data from the source to the destination. A 0 inhibits data from passing from the source to the destination.

Entering Parameters

For all micro controllers, specify I0. For 16 I/O controllers, I0/0–9 are valid and I0/10–15 are considered unused inputs (they do not physically exist). For 32 I/O controllers, I0/0–15 and I1/0–3 are valid. Specify I1 if you want to immediately update the last four input bits.

Mask – Specify a Hex constant or register address.

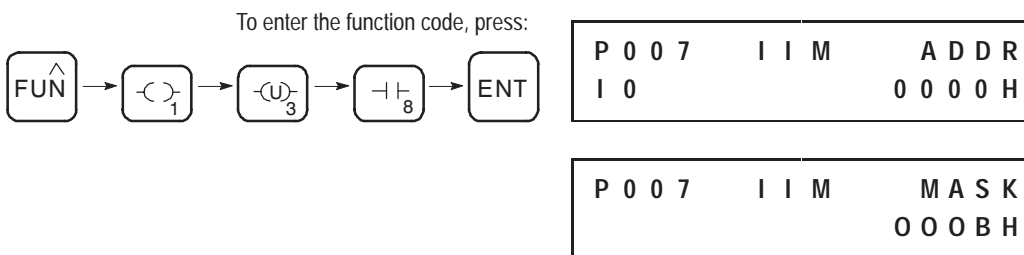
Entering the Instruction

You enter the instruction from within the program monitor functional area. The following items apply when entering the instruction:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:

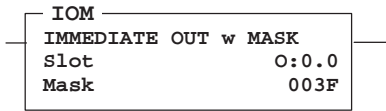


Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17–4.)



Immediate Output with Mask (IOM)

Ladder representation:



Execution Times (μsec) when:

| True | False |
|-------|-------|
| 41.59 | 6.78 |

This instruction allows you to update the outputs prior to the normal output scan. Data from the output image is transferred through a mask to the specified outputs. The program scan then resumes.

Entering Parameters

For all micro controllers, specify 00. For 16 I/O controllers, 00/0–5 are valid and 00/6–15 are considered unused inputs (they do not physically exist). For 32 I/O controllers, 00/0–11 are valid and 00/12–15 are unused.

Mask – Specify a Hex constant or register address.

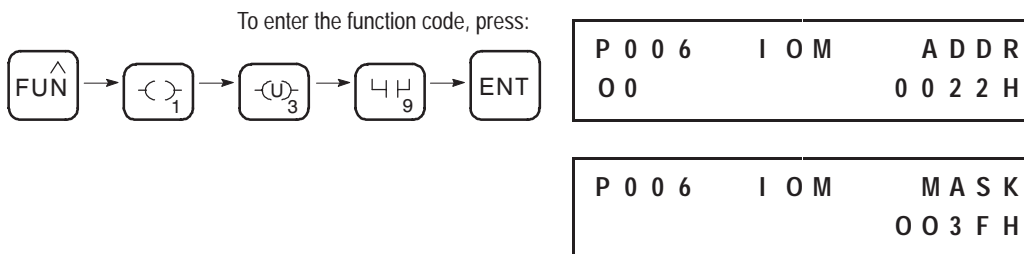
Entering the Instruction

You enter the instruction from within the program monitor functional area. The following items apply when entering the instruction:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17-4.)



Program Flow Control Instructions in the Paper Drilling Machine Application Example

To demonstrate the use of program flow control instructions, this section provides ladder rungs followed by the optimized instruction list for these rungs. The rungs are part of the paper drilling machine application example described in appendix E. You will be adding to the main program in file 2. The new rungs are needed to call the other subroutines containing the logic necessary to run the machine.

Ladder Rungs

Rung 2:5
Calls the drill sequence subroutine. This subroutine manages the operation of a drilling sequence and restarts the conveyor upon completion of the drilling sequence.

```

|-----+JSR-----+
|-----+JUMP TO SUBROUTINE+-
|-----|SBR file number 6|
|-----+-----+

```

Rung 2:6
Calls the subroutine that tracks the amount of wear on the current drill bit.

```

|-----+JSR-----+
|-----+JUMP TO SUBROUTINE+-
|-----|SBR file number 7|
|-----+-----+

```

Rung 2:7

```

|-----+END+-----|

```

Instruction List

File 2, Rung 5

Calls the drill sequence subroutine. This subroutine manages the operation of a drilling sequence and restarts the conveyor upon completion of the drilling sequence.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| 132 | | JSR | SBR# | 6 | |

File 2, Rung 6

Calls the subroutine that tracks the amount of wear on the current drill bit.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| 132 | | JSR | SBR# | 7 | |

Using Application Specific Instructions

This chapter contains general information about the application specific instructions and explains how they function in your application program. Each of the instructions includes information on:

- what the instruction symbol looks like
- typical execution time for the instruction
- how to use the instruction
- how to enter the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the application specific instructions in use.

Application Specific Instructions

| HHP Display | Mnemonic | Function Code | Name | Purpose | Page |
|-------------|----------|---------------|------------------------------------|--|-------|
| | BSL | 150 | Bit Shift Left | Loads a bit of data into a bit array, shifts the pattern of data to the left through the array, and unloads the last bit of data in the array. | 13-3 |
| | BSR | 151 | Bit Shift Right | Loads a bit of data into a bit array, shifts the pattern of data to the right through the array, and unloads the last bit of data in the array. | 13-4 |
| | SQO | 152 | Sequencer Output | Control sequential machine operations by transferring 16-bit data through a mask to image addresses. | 13-6 |
| | SQC | 153 | Sequencer Compare | | |
| | SQL | 154 | Sequencer Load | Capture referenced conditions by manually stepping the machine through its operating sequences. | 13-12 |
| | STD | 155 | Selectable Timer Interrupt Disable | Output instructions, associated with the Selectable Timed Interrupt function. STD and STE are used to prevent an STI from occurring during a portion of the program. | 13-17 |
| | STE | 156 | Selectable Timer Interrupt Enable | | |
| | STS | 157 | Selectable Timer Interrupt Start | Initiates a Selectable Timed Interrupt. | 13-18 |
| ┌ INT ─┘ | LD INT | 158 | Interrupt Subroutine | Associated with Selectable Timed Interrupts or HSC Interrupts. | 13-19 |

About the Application Specific Instructions

These instructions simplify your program by allowing you to use a single instruction or pair of instructions to perform common complex operations.

Since these are *output* instructions (except LD INT), they do not have LD, AND, and OR equivalents.

In this chapter, you will find a general overview preceding groups of instructions. Before you learn about the instructions in each of these groups, we suggest that you read the overview. This chapter contains the following overviews:

- Bit Shift Instructions Overview
- Sequencer Instructions Overview
- Selectable Timed Interrupt (STI) Function Overview

Bit Shift Instructions Overview

The following general information applies to bit shift instructions.

Entering Parameters

Enter the following parameters when programming these instructions:

- **File** is the address of the bit array you want to manipulate. You must use the file indicator (#) in the bit array address. (The HHP inserts the # character automatically.)
- **Control** is the address of the control element that stores the status byte of the instruction, the size of the array (in number of bits). Note that the control address should not be used for any other instruction.

The control element is shown below.

| | 15 | 13 | 11 | 10 | 00 |
|--------|------------------------------------|----|----|----|----------|
| Word 0 | EN | DN | ER | UL | Not used |
| Word 1 | Size of bit array (number of bits) | | | | |
| Word 2 | Reserved | | | | |

Status bits of the control element may be addressed by mnemonic. They include:

- **Unload Bit UL** (bit 10) is the instruction's output.
- **Error Bit ER** (bit 11), when set, indicates the instruction detected an error such as entering a negative number for the length or position. Avoid using the unload bit when this bit is set.
- **Done Bit DN** (bit 13), when set, indicates the bit array shifted one position.
- **Enable Bit EN** (bit 15) is set on a false-to-true transition of the rung and indicates the instruction is enabled.

When the register shifts and input conditions go false, the enable, done, and error bits are reset.

- **Bit Address** is the address of the source bit. The status of this bit is inserted in either the first (lowest) bit position (BSL) or last (highest) bit position (BSR).
- **Length** (size of bit array) is the number of bits in the bit array, up to 1680 bits. A length value of 0 causes the input bit to be transferred to the UL bit.

A length value that points past the end of the programmed file causes a major error to occur. *If you alter a length value with your program, make certain that the altered value is valid.*

The instruction invalidates all bits beyond the last bit in the array (as defined by the length) up to the next word boundary.

Entering the Instructions

The following items apply when entering the instructions:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).

- You can return to previously entered operands by pressing this key:



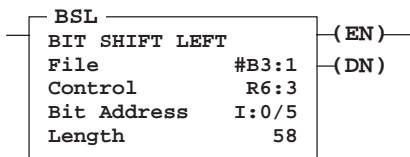
Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17-4.)

Effects on Index Register S24

The shift operation clears the index register S24 to zero.

Bit Shift Left (BSL)

Ladder representation:



When the rung goes from false-to-true, the controller sets the enable bit (EN bit 15) and the data block is shifted to the left (to a higher bit number) one bit position. The specified bit at the bit address is shifted into the first bit position. The last bit is shifted out of the array and stored in the unload bit (UL bit 10). The shift is completed immediately.

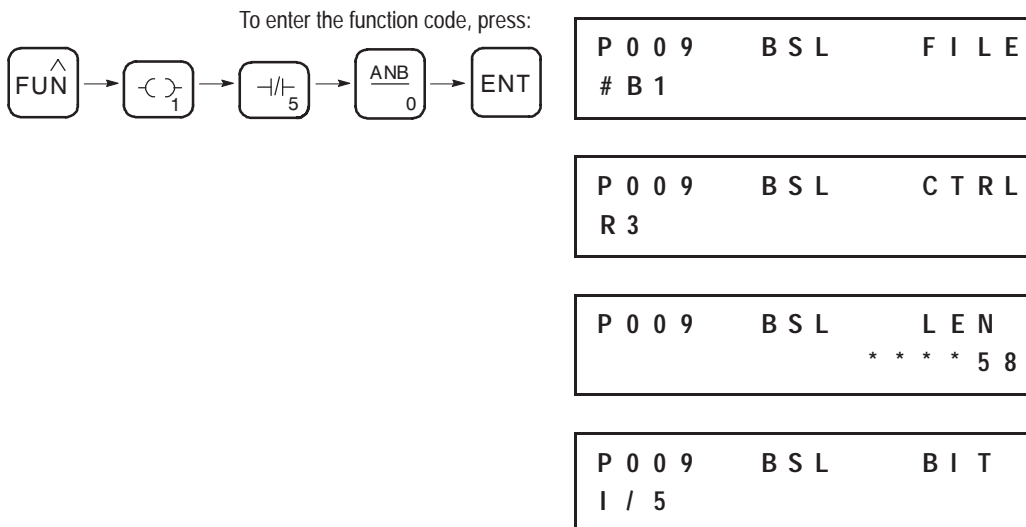
For wraparound operation, set the bit address to the last bit of the array or to the UL bit.

Execution Times (μsec) when:

| | True | False |
|--------------------------------|------|-------|
| 53.71+5.24 x position value | | 19.80 |

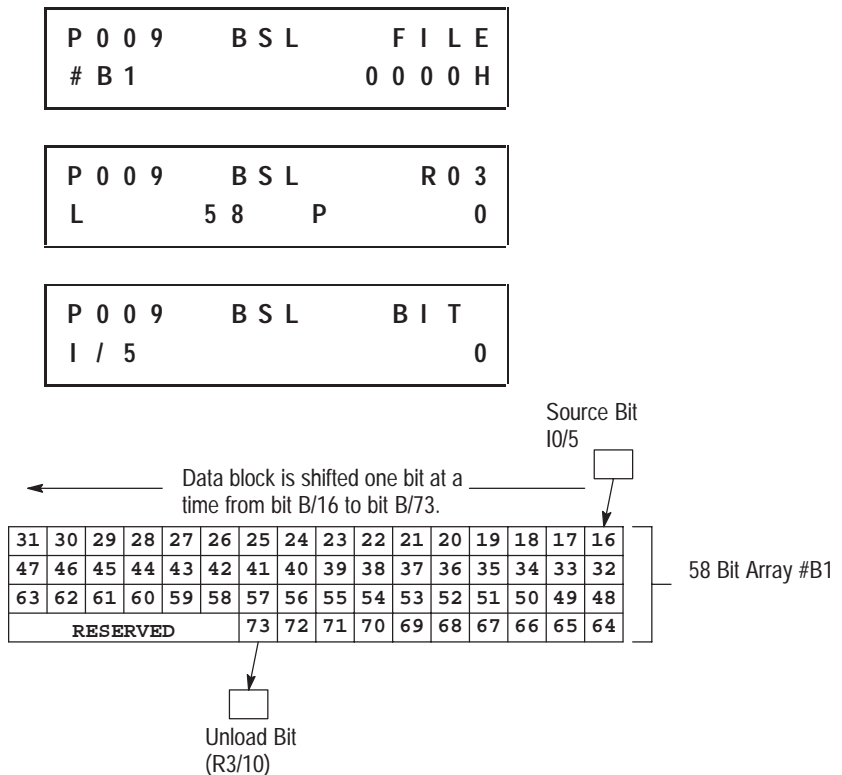
Entering the Instruction

You enter the instruction from within the program monitor functional area.



Operation

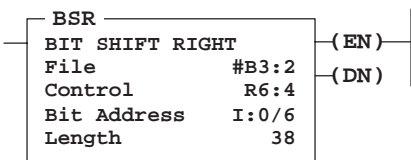
The operation of the BSL instruction is shown in the figure below. The screens shown above the figure are the condensed screens that appear after instruction entry is complete.



If you wish to shift more than one bit per scan, you must create a loop in your application using the JMP, LBL, and CTU instructions.

Bit Shift Right (BSR)

Ladder representation:



Execution Times (µsec) when:

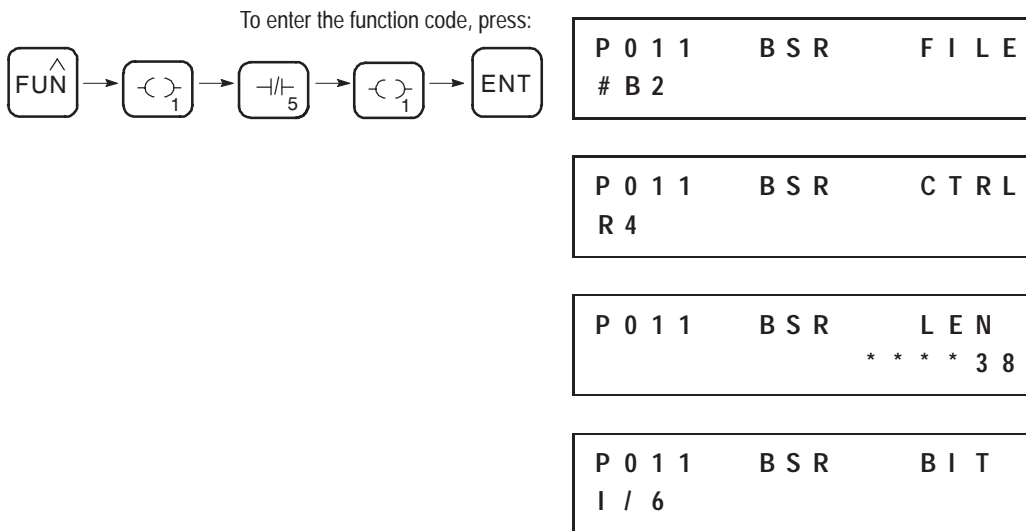
| | True | False |
|-----------------------------|------|-------|
| 53.34+3.98 x position value | | 19.80 |

When the rung goes from false-to-true, the controller sets the enable bit (EN bit 15) and the data block is shifted to the right (to a lower bit number) one bit position. The specified bit at the bit address is shifted into the last bit position. The first bit is shifted out of the array and stored in the unload bit (UL bit 10). The shift is completed immediately.

For wraparound operation, set the bit address to the first bit of the array or to the UL bit.

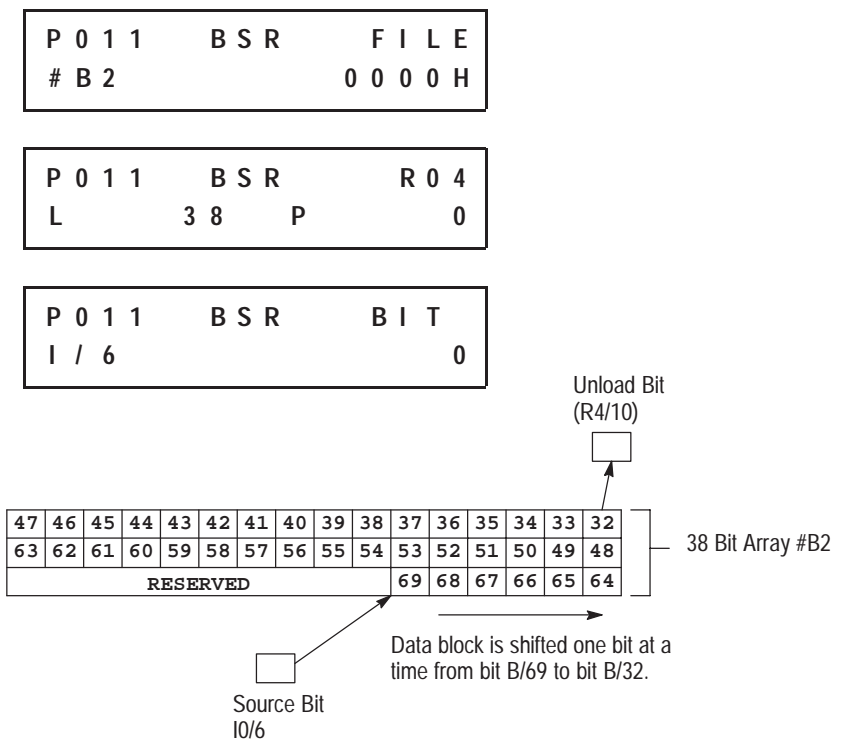
Entering the Instruction

You enter the instruction from within the program monitor functional area.



Operation

The operation of the BSR instruction is shown in the figure below. The screens shown above the figure are the condensed screens that appear after instruction entry is complete.



If you wish to shift more than one bit per scan, you must create a loop in your application using the JMP, LBL, and CTU instructions.

Sequencer Instructions Overview

The following general information applies to sequencer instructions.

Entering the Instructions

The following items apply when entering the instructions:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17–4.)

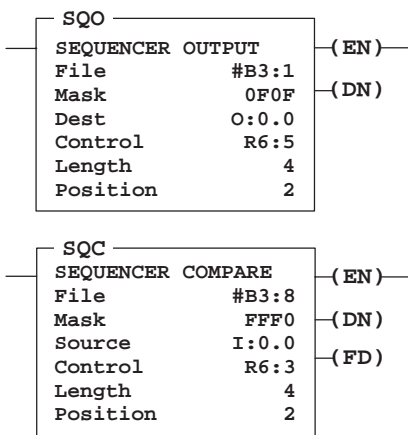
Effects on Index Register S24

The value present in the index register S24 is overwritten when the sequencer instruction is true. The index register value will equal the position value of the instruction.

Sequencer Output (SQO) and Sequencer Compare (SQC)

These instructions transfer 16-bit data to word addresses for the control of sequential machine operations.

Ladder representation:



Execution Times (μsec) when:

| | True | False |
|-----|-------|-------|
| SQO | 60.52 | 27.40 |
| SQC | 60.52 | 27.40 |

Entering Parameters

Enter the following parameters when programming these instructions:

- **File** is the address of the sequencer file. You must use the file indicator (#) for this address. (The HHP inserts the # character automatically.)

Sequencer file data is used as follows:

| Instruction | Sequencer File Stores |
|-------------|--------------------------------------|
| SQO | Data for controlling outputs |
| SQC | Reference data for monitoring inputs |

- **Mask** (SQO, SQC) is a hexadecimal code or the address of the mask word or file through which the instruction moves data. Set mask bits to pass data and clear mask bits to prevent the instruction from operating or corresponding destination bits. Use a mask word or file if you want to change the mask according to application requirements.

If the mask is a file, its length will be equal to the length of the sequencer file. The two files track automatically.

- **Source** is the address of the input word or file for a SQC from which the instruction obtains data for comparison to its sequencer file.

- **Destination** is the address of the output word or file for a SQO to which the instruction moves data from its sequencer file.

Important: You can address the mask, source, or destination of a sequencer instruction as a word or file. If you address it as a file (using file indicator #), the instruction automatically steps through the source, mask, or destination file.

- **Control (SQO, SQC)** is the control structure that stores the status byte of the instruction, the length of the sequencer file, and the current position in the file. You should not use the control address for any other instruction.

| | 15 | 13 | 11 | 08 | 00 |
|--------|--------------------------|----|----|----|----|
| Word 0 | EN | DN | ER | FD | |
| Word 1 | Length of sequencer file | | | | |
| Word 2 | Position | | | | |

Status bits of the control structure include:

- **Found Bit FD** (bit 08) – SQC only. When the status of all non-masked bits in the source address match those of the corresponding reference word, the FD bit is set. This bit is assessed each time the SQC instruction is evaluated while the rung is true.
 - **Error Bit ER** (bit 11) is set when the controller detects a negative position value, or a negative or zero length value. When the ER bit is set, the minor error bit (S2) is also set. Both bits must be cleared.
 - **Done Bit DN** (bit 13) is set by the SQO or SQC instruction after it has operated on the last word in the sequencer file. It is reset on the next false-to-true rung transition after the rung goes false.
 - **Enable EN** (bit 15) is set by a false-to-true rung transition and indicates the SQO or SQC instruction is enabled.
- **Length** is the number of steps of the sequencer file starting at position 1. The maximum number you can enter is 104 words. Position 0 is the startup position. The instruction resets (wraps) to position 1 at each cycle completion.
 - **Position** is the word location or step in the sequencer file from/to which the instruction moves data.

Tip

You may use the RES instruction to reset a sequencer. All control bits (except FD) are reset to zero. The position is also set to zero. Program the address of your control register in the RES (e.g., R0).

Using SQO

This output instruction steps through the sequencer file whose bits have been set to control various output devices.

When the rung goes from false-to-true, the instruction increments to the next step (word) in the sequencer file. Data stored there is transferred through a mask to the destination address specified in the instruction. Data is written to the destination word every time the instruction is executed.

The done bit is set when the last word of the sequencer file is transferred. On the next false-to-true rung transition, the instruction resets the position to step one.

If the position is equal to zero at startup, when you switch the controller from the RPRG mode to the RRUN mode, instruction operation depends on whether the rung is true or false on the first scan.

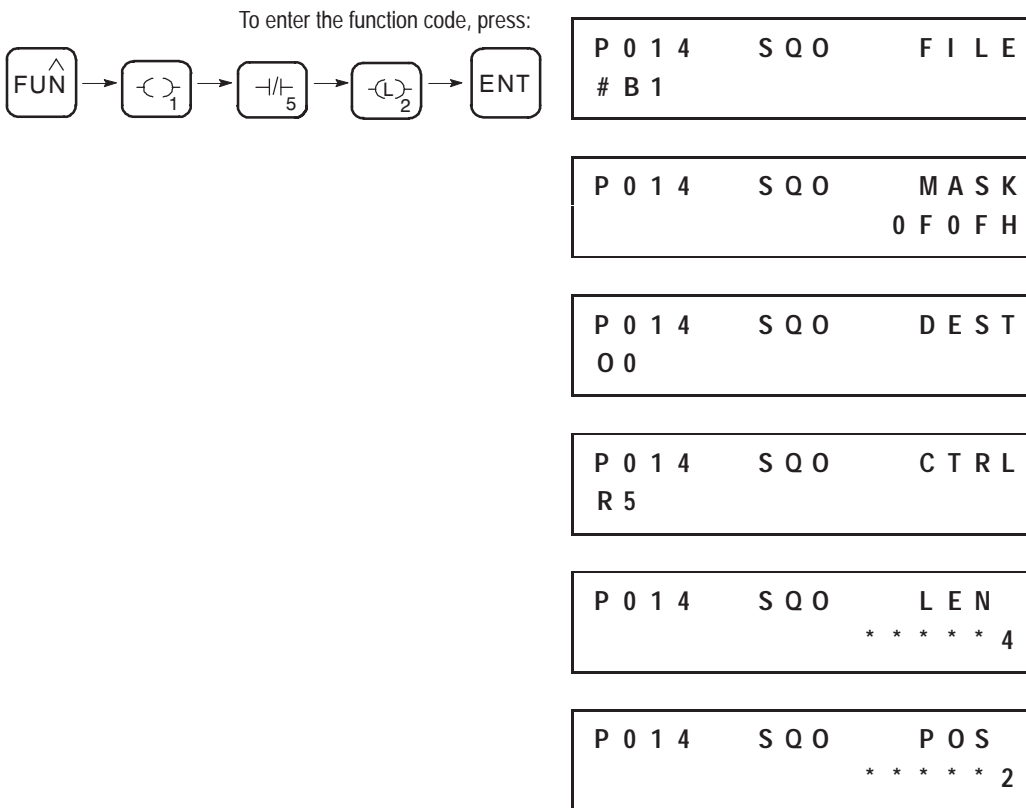
- If true, the instruction transfers the value in step zero.
- If false, the instruction waits for the first rung transition from false-to-true and transfers the value in step one.

The bits mask data when reset and pass data when set. The instruction will not change the value in the destination word unless you set mask bits.

The mask can be fixed or variable. It will be fixed if you enter a hexadecimal code. It will be variable if you enter an element address or a file address for changing the mask with each step.

Entering the Instruction

You enter the instruction from within the program monitor functional area.



Operation

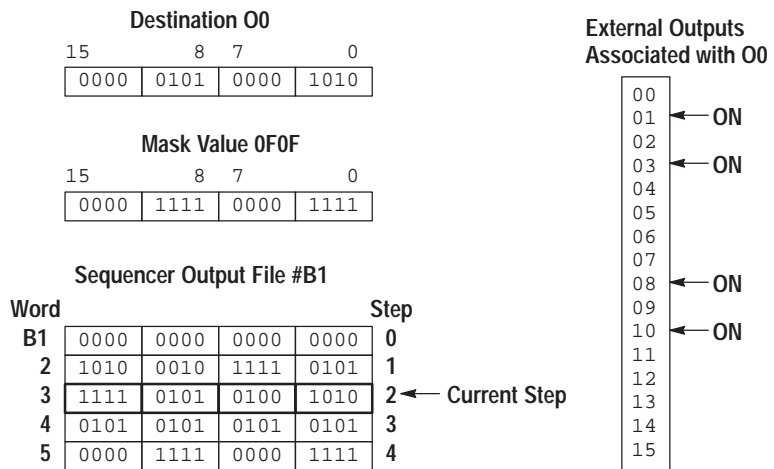
The operation of the SQO instruction is shown in the figure below. The screens shown above the figure are the condensed screens that appear after instruction entry is complete.

| | | |
|---------|-------|-----------|
| P 0 1 4 | S Q O | F I L E |
| # B 1 | | 0 0 0 0 H |

| | | |
|---------|-------|-----------|
| P 0 1 4 | S Q O | M A S K |
| | | 0 F 0 F H |

| | | |
|---------|-------|-----------|
| P 0 1 4 | S Q O | D E S T |
| 0 0 | | 0 0 2 2 H |

| | | |
|---------|-------|-------|
| P 0 1 4 | S Q O | R 0 5 |
| L | 4 P | 2 |



Using SQC

When the status of all non-masked bits in the source word match those of the corresponding reference word, the instruction sets the found bit (FD) in the control word. Otherwise, the found bit (FD) is cleared.

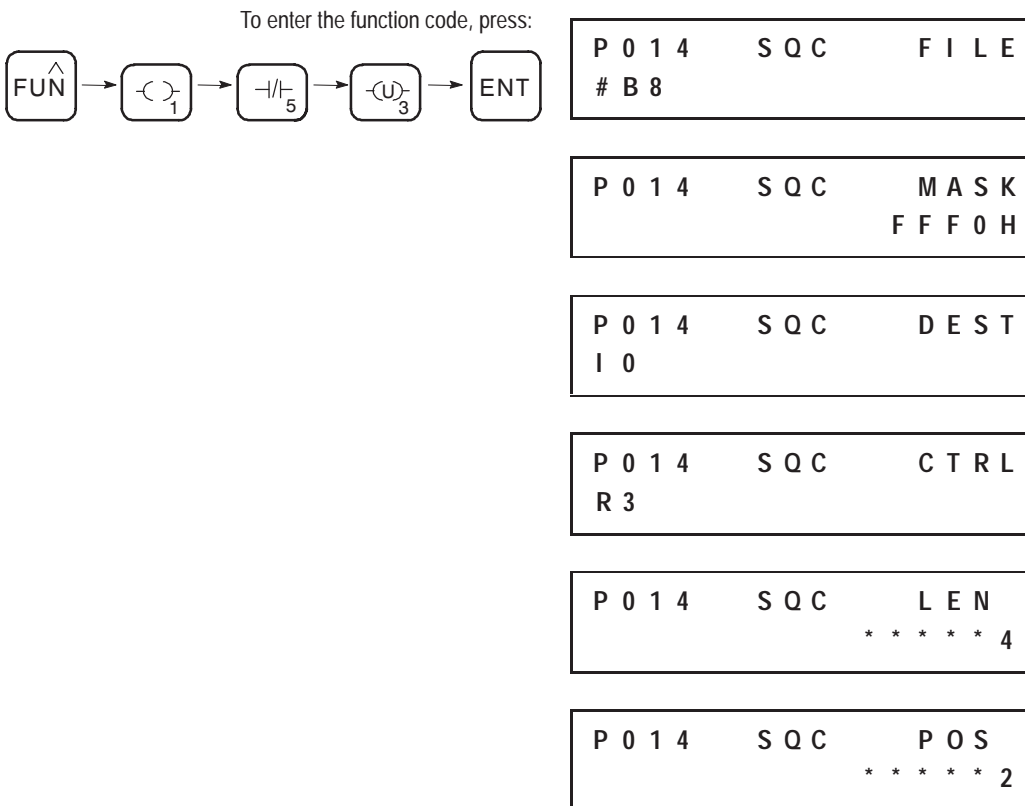
The bits mask data when reset and pass data when set.

The mask can be fixed or variable. If you enter a hexadecimal code, it is fixed. If you enter an element address or a file address for changing the mask with each step, it is variable.

When the rung goes from false-to-true, the instruction increments to the next step (word) in the sequencer file. Data stored there is transferred through a mask and compared against the source for equality. While the rung remains true, the source is compared against the reference data for every scan. If equal, the FD bit is set in the SQCs control counter.

Entering the Instruction

You enter the instruction from within the program monitor functional area.



Operation

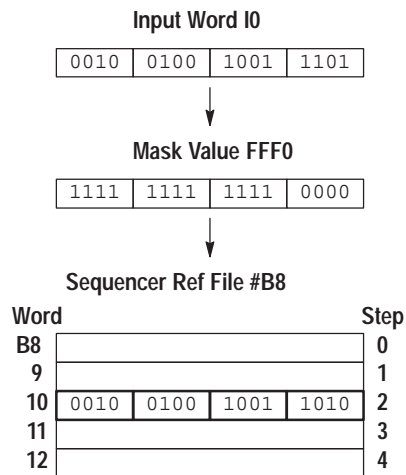
The operation of the SQC instruction is shown in the figure below. The screens shown above the figure are the condensed screens that appear after instruction entry is complete.

| | | |
|---------|-------|-----------|
| P 0 1 4 | S Q C | F I L E |
| # B 8 | | 0 0 0 0 H |

| | | |
|---------|-------|-----------|
| P 0 1 4 | S Q C | M A S K |
| | | F F F 0 H |

| | | |
|---------|-------|-----------|
| P 0 1 4 | S Q C | S R C |
| I 0 | | 0 0 0 0 H |

| | | |
|---------|-------|-------|
| P 0 1 4 | S Q C | R 0 3 |
| L | 4 P | 2 |

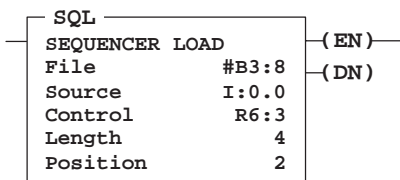


The SQC FD bit is set when the instruction detects that an input word matches (thru mask) its corresponding reference word.

The FD bit R3/FD is set in this example, since the input word matches the sequencer reference value using the mask value.

Sequencer Load (SQL)

Ladder representation:



Execution Times (μ sec) when:

| True | False |
|-------|-------|
| 53.41 | 28.12 |

The SQL instruction stores 16-bit data into a sequencer load file at each step of sequencer operation. The source of this data can be an I/O or internal word address, a file address, or a constant.

Entering Parameters

Enter the following parameters when programming this instruction:

- **File** is the address of the sequencer file. You must use the file indicator (#) for this address. (The HHP inserts the # character automatically.)
- **Source** can be a word address, file address, or a constant (–32,768 to 32,767).
If the source is a file address, the file length equals the length of the sequencer load file. The two files step automatically, according to the position value.
- **Length** is the number of steps of the sequencer load file (and also of the source if the source is a file address), starting at position 1. The maximum number you can enter is 104 words. Position 0 is the startup position. The instruction resets (wraps) to position 1 at each cycle completion.
- **Position** is the word location or step in the sequencer file to which data is moved.
- **Control** is a control file address. The status bits, length value, and position value are stored in this element. Do not use the control file address for any other instruction.

The control element is shown below:

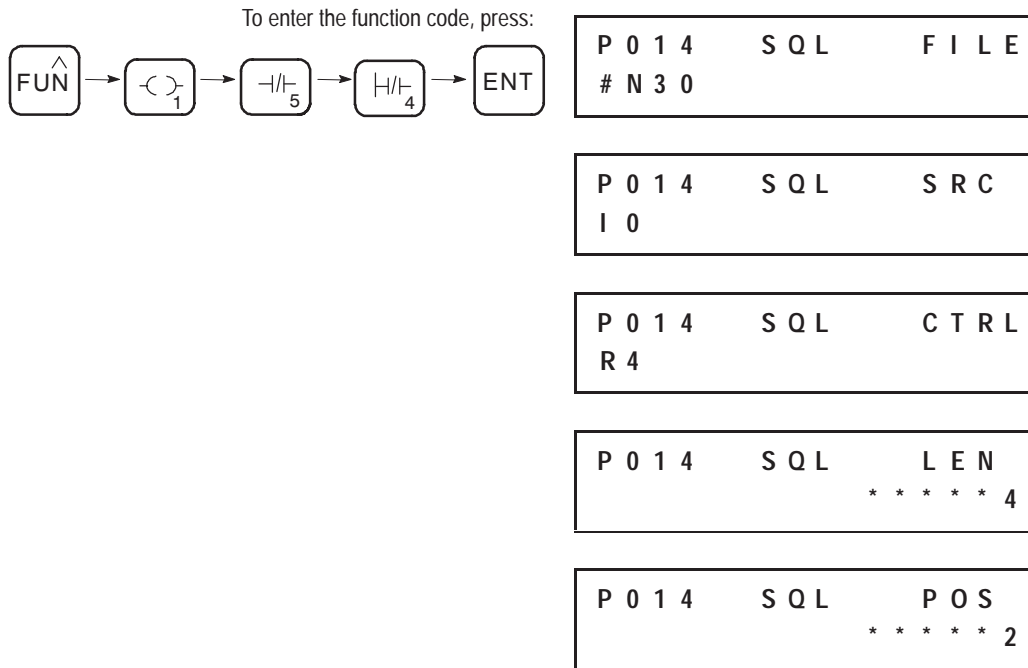
| | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|--------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Word 0 | EN | | DN | | ER | | | | | | | | | | | |
| Word 1 | Length | | | | | | | | | | | | | | | |
| Word 2 | Position | | | | | | | | | | | | | | | |

Status bits of the control structure include:

- **Error Bit ER** (bit 11) is set when the controller detects a negative position value or a negative or zero length value. When the ER bit is set, the minor error bit (S2) is also set. Both bits must be cleared.
- **Done Bit DN** (bit 13) is set after the instruction has operated on the last word in the sequencer load file. It is reset on the next false-to-true rung transition after the rung goes false.
- **Enable Bit EN** (bit 15) is set on a false-to-true transition of the SQL rung and reset on a true-to-false transition.

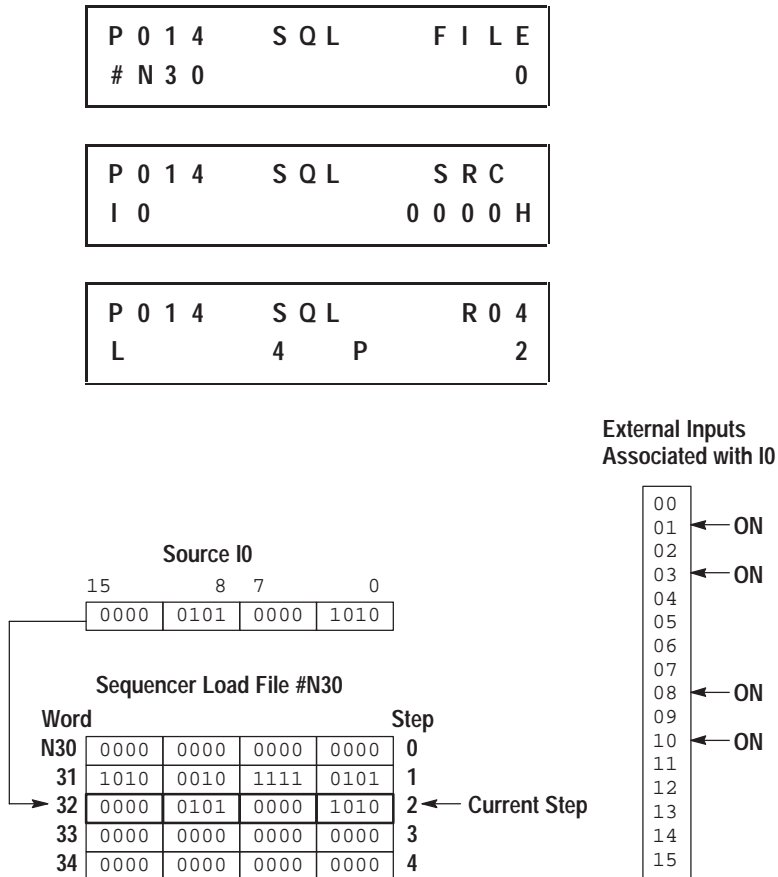
Entering the Instruction

You enter the instruction from within the program monitor functional area.



Operation

The operation of the SQL instruction is shown in the figure below. The screens shown above the figure are the condensed screens that appear after instruction entry is complete. Input word I0 is the source. Data in this word is loaded into integer file #N30 by the sequencer load instruction.



When rung conditions change from false-to-true, the SQL enable bit (EN) is set. The control element R4 increments to the next position in the sequencer file and loads the contents of source I0 into the corresponding location in the file. The SQL instruction continues to load the current data into this location each scan that the rung remains true. When the rung goes false, the enable bit (EN) is reset.

The instruction loads data into a new file element at each false-to-true transition of the rung. When step 4 is completed, the done bit (DN) is set. Operation cycles to position 1 at the next false-to-true transition of the rung after position 4.

If the source were a file address such as #N40, files #N40 and #N30 would both have a length of 5 (0-4) and would track through the steps together per the position value.

Selectable Timed Interrupt (STI) Function Overview

The Selectable Timed Interrupt (STI) function allows you to interrupt the scan of the application program automatically, on a periodic basis, to scan a subroutine file. Afterwards, the controller resumes executing the application program from the point where it was interrupted.

Basic Programming Procedure for the STI Function

To use the STI function in your application file:

1. Enter the desired rungs in File 5. (File 5 is designated for the STI subroutine.)
2. Enter the setpoint (the time between successive interrupts) using the program configuration option of the HHP's menu. (See page 18–9.) The range is 10–2550 ms (entered in 10 ms increments). A setpoint of zero disables the STI function.

Important: The setpoint value must be a longer time than the execution time of the STI subroutine file, or a minor error bit is set.

Operation

After you restore your program and enter the RRUN, RCSN, or RSSN mode, the STI begins operation as follows:

1. The STI timer begins timing.
2. When the STI interval expires, the program scan is interrupted and the STI subroutine file is scanned; the STI timer is reset.
3. If while executing the STI (file 5), another STI interrupt occurs the STI pending bit (S2/0) is set.
4. If while an STI is pending the STI timer expires, the STI lost bit (S5/10) is set.
5. When the STI subroutine scan is completed, scanning of the program resumes at the point where it left off, unless an STI is pending. In this case the subroutine is immediately scanned again.
6. The cycle repeats.

For identification of your STI subroutine, include an INT instruction as the first instruction on the first rung of the file.

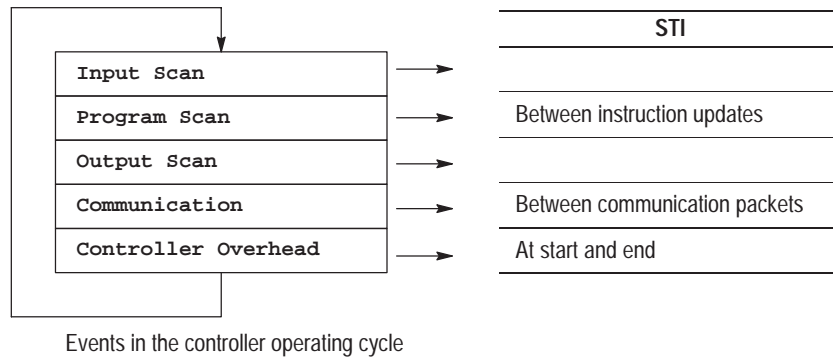
STI Subroutine Content

The STI subroutine contains the rungs of your application logic. You can program any instruction inside the STI subroutine except a TND instruction. IIM or IOM instructions are needed in an STI subroutine if your application requires immediate update of input or output points. End the STI subroutine with an RET instruction.

JSR stack depth is limited to three. You may call other subroutines to a level three deep from an STI subroutine.

Interrupt Latency and Interrupt Occurrences

Interrupt latency is the interval between the STI timeout and the start of the interrupt subroutine. STI interrupts can occur at any point in your program, but not necessarily at the same point on successive interrupts. The table below shows the interaction between an interrupt and the controller operating cycle.



Note that STI execution time adds directly to the overall scan time. During the latency period, the controller is performing operations that cannot be disturbed by the STI interrupt function.

Interrupt Priorities

Interrupt priorities are as follows:

1. User Fault Routine
2. High-Speed Counter
3. Selectable Timed Interrupt

An executing interrupt can only be interrupted by an interrupt having a higher priority.

Status File Data Saved

Data in the following words is saved on entry to the STI subroutine and re-written upon exiting the STI subroutine.

- S0 Arithmetic flags
- S13 and S14 Math register
- S24 Index register

Selectable Timed Disable (STD) and Enable (STE)

Ladder representation:



Execution Times (μsec) when:

| | True | False |
|-----|-------|-------|
| STD | 6.69 | 3.16 |
| STE | 10.13 | 3.16 |

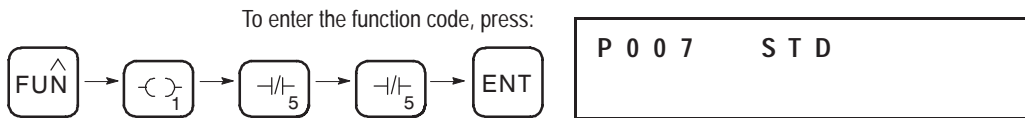
These instructions are generally used in pairs. The purpose is to create zones in which STI interrupts *cannot* occur.

Using STD

When true, this instruction resets the STI enable bit and prevents the STI subroutine from executing. When the rung goes false, the STI enable bit remains reset until a true STD or STE instruction is executed. The STI timer continues to operate while the enable bit is reset.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

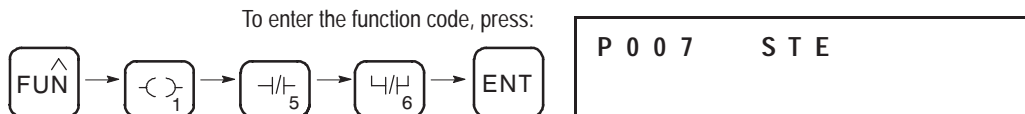


Using STE

This instruction sets the STI enable bit and allows execution of the STI subroutine. When the rung goes false, the STI enable bit remains set until a true STD instruction is executed. This instruction has no effect on the operation of the STI timer or setpoint. When the enable bit is set, the first execution of the STI subroutine can occur at any point up to the full STI interval.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

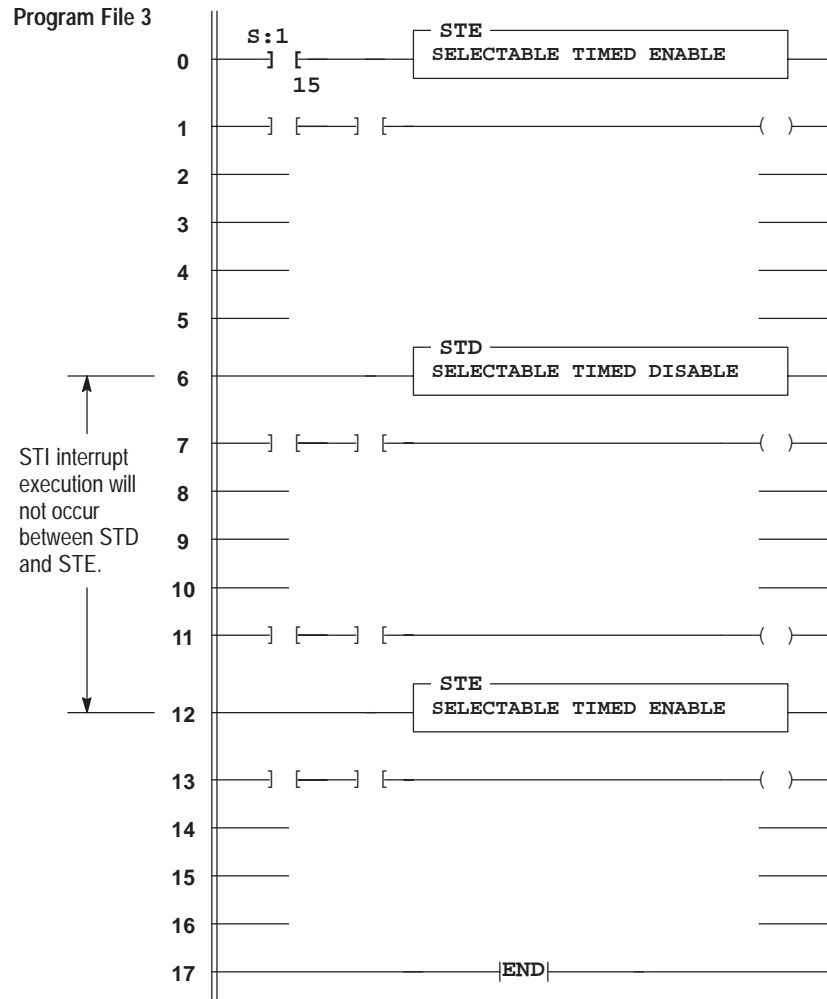


STD/STE Zone Example

In the program that follows, the STI function is in effect. The STD and STE instructions in rungs 6 and 12 are included in the program to avoid having STI subroutine execution at any point in rungs 7 through 11.

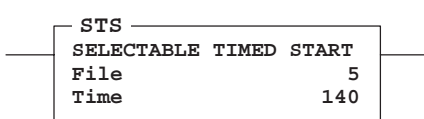
The STD instruction (rung 6) resets the STI enable bit and the STE instruction (rung 12) sets the enable bit again. The STI timer increments and may time out in the STD zone, setting the pending bit S2/0 and lost bit S5/10.

The first pass bit S1/15 and the STE instruction in rung 0 are included to insure that the STI function is initialized following a power cycle. You should include this rung any time your program contains an STD/STE zone or an STD instruction.



Selectable Timed Start (STS)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 24.59 | 6.78 |

Use the STS instruction to condition the start of the STI timer upon entering the RRUN, RCSN, or RSSN mode – rather than starting automatically. You can also use it to set up or change setpoint/frequency of the STI routine that is executed when the STI timer expires.

This instruction is not required to configure a basic STI interrupt application.

The STS instruction requires you to enter the parameter for the STI setpoint using the HHP’s program configuration menu option. Upon a true execution of the rung, this instruction enters the setpoint in the status file (S30), overwriting the existing data. At the same time, the STI timer is reset and begins timing; at timeout, the STI subroutine execution occurs. When the rung goes false, the STI function remains enabled at the setpoint you’ve entered in the STS instruction.

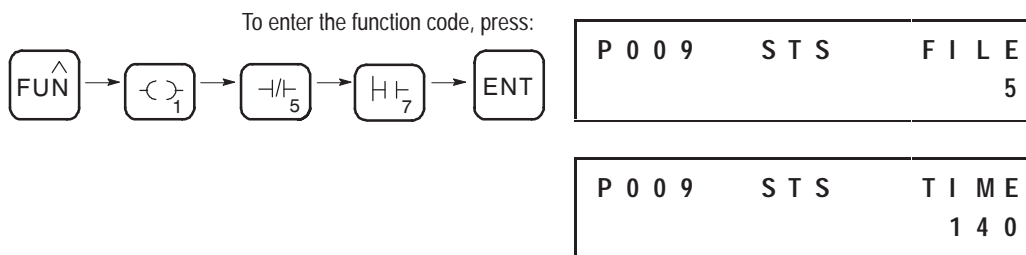
Entering the Instruction

You enter the instructions from within the program monitor functional area. The following items apply when entering the instructions:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:

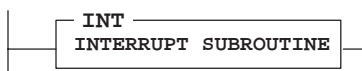


Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters you must go into the overwrite mode. (See page 17-4.)



Interrupt Subroutine (INT)

Ladder representation:



Execution Times (μ sec) when:

| True | False |
|------|-------|
| 1.45 | 0.99 |

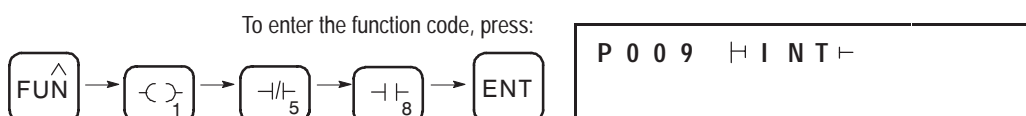
This instruction serves as a label or identifier of a program file as an interrupt subroutine (INT label) versus a regular subroutine (SBR label).

This instruction has no control bits and is always evaluated as true. Use of this instruction is optional; however, we recommend using it.

Important: The instruction must be programmed as the first instruction of the first rung of a subroutine. (Since it must be the first instruction on the rung, the INT instruction is also known as LD INT.)

Entering the Instruction

You enter the instruction from within the program monitor functional area.

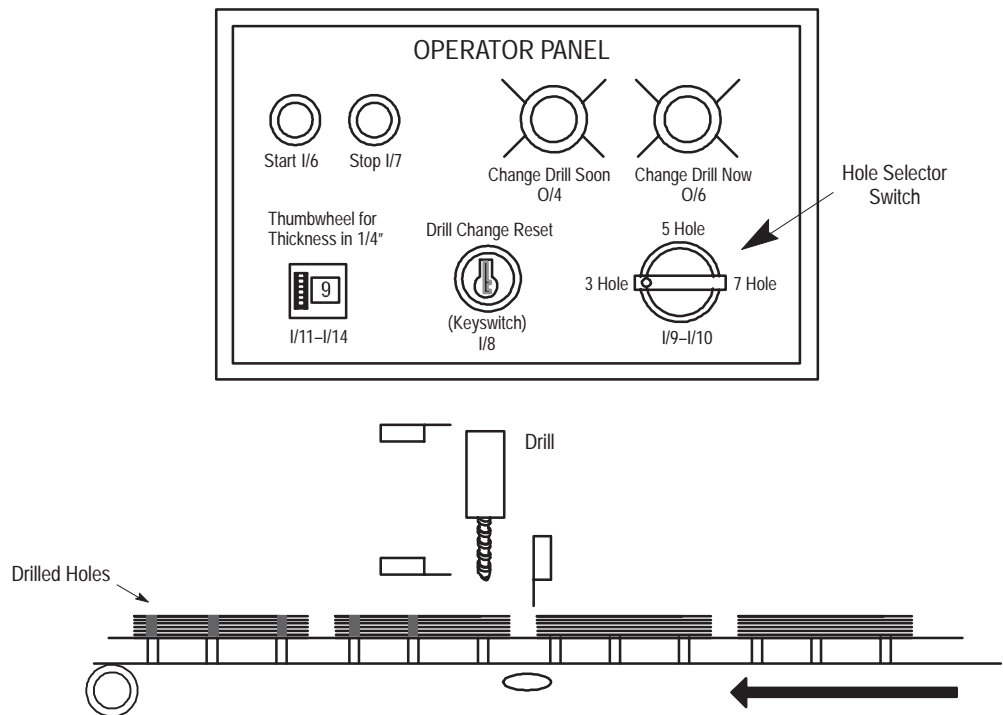


Application Specific Instructions in the Paper Drilling Machine Application Example

To demonstrate the use of application specific instructions, this section provides ladder rungs followed by the optimized instruction list of these rungs. The rungs are part of the paper drilling machine application example described in appendix E. You will begin a subroutine in file 4.

This portion of the subroutine tells the conveyor where to stop to allow a hole to be drilled. The stop positions are different for each hole pattern (3-hole, 5-hole, 7-hole), so separate sequencers are used to store and access each of the three hole patterns.

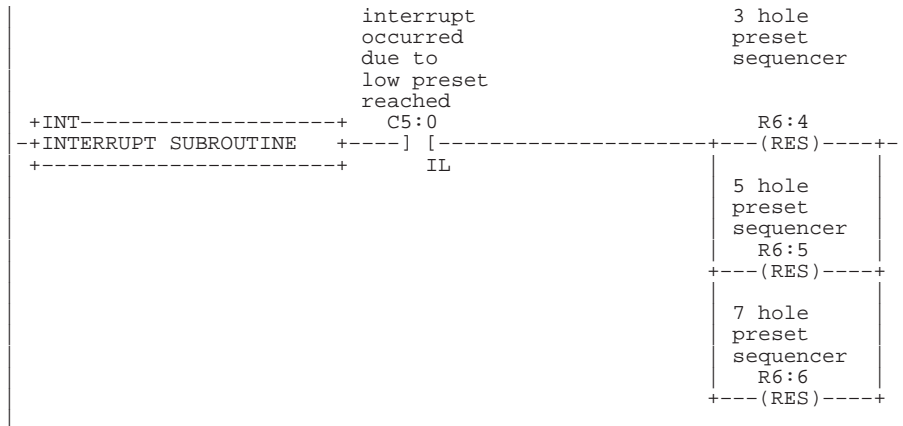
Important: If you use a 16 I/O controller, only the 5 hole drill pattern can be used.



Ladder Rungs

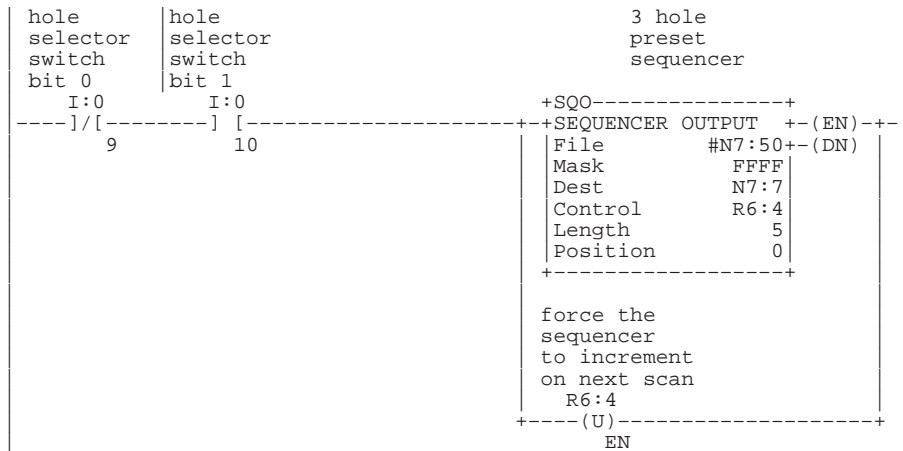
Rung 4:0

Resets the hole count sequencers each time the low preset is reached. The low preset has been set to zero to cause an interrupt to occur each time that a reset occurs. The low preset is reached anytime that a reset C5:0 or hardware reset occurs. This ensures that the first preset value is loaded into the HSC at each entry into the RRUN mode and each time that the external reset signal is activated.



Rung 4:1^①

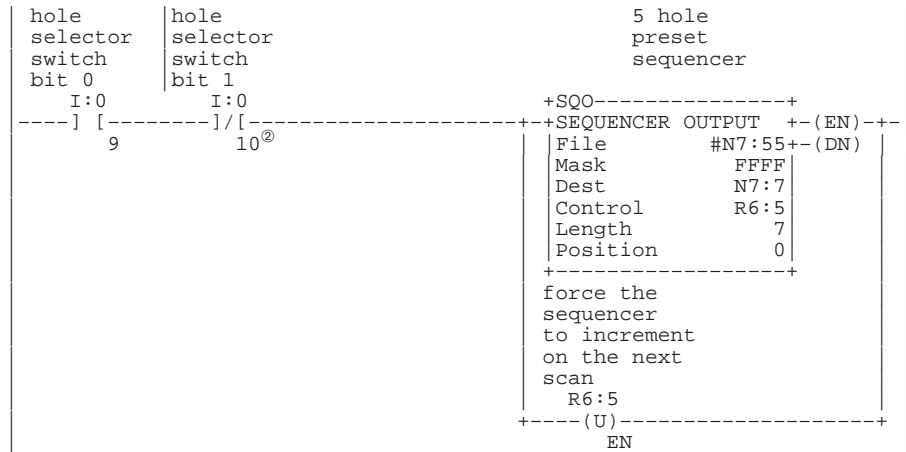
Keeps track of the hole number that is being drilled and loads the correct HSC preset based on the hole count. This rung is only active when the "hole selector switch" is in the "3-hole" position. The sequencer uses step 0 as a null step upon reset. It uses the last step as a "go forever" in anticipation of the "end of manual" hard-wired external reset.



^① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.

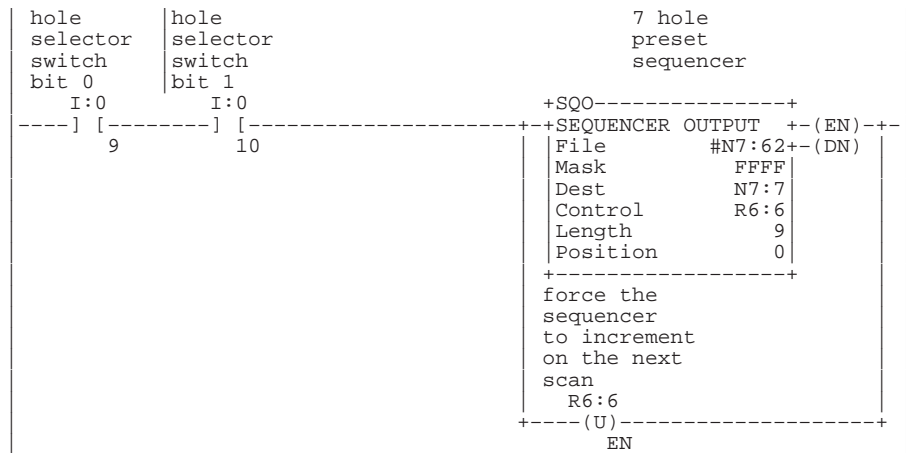
Rung 4:2

Is identical to the previous rung except that it is only active when the "hole selector switch" is in the "5-hole" position.



Rung 4:3^{①③}

Is identical to the two previous rungs except that it is only active when the "hole selector switch" is in the "7-hole" position.



- ① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.
- ② This instruction accesses I/O only available with 32 I/O controllers. Therefore, do not include this instruction if you are using a 16 I/O controller.
- ③ More rungs will be added to this subroutine at the end of chapter 14.

Instruction List

File 4, Rung 0

Resets the hole count sequencers each time that the low preset is reached. The low preset has been set to zero to cause an interrupt to occur each time that a reset occurs. The low preset is reached anytime that a reset C5:0 or hardware reset occurs. This ensures that the first preset value is loaded into the high-speed counter at each entry into the RRUN mode and each time that the external reset signal is activated.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|-------|--------|
| 158 | -INT- | LD-INT | | | |
| 22 | -] [- | AND | interrupt occurred due to low preset reached C0/IL | 0 | |
| 7 | | RES | 3 hole preset sequencer R4 | | |
| 7 | | RES | 5 hole preset sequencer R5 | | |
| 7 | | RES | 7 hole preset sequencer R6 | | |

File 4, Rung 1^①

Keeps track of the hole number that is being drilled and loads the correct high-speed counter preset based on the hole count. This rung is only active when the "hole selector switch" is in the "3-hole" position. The sequencer uses step 0 as a null step upon reset. It uses the last step as a "go forever" in anticipation of the "end of manual" hard-wired external reset.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|-------|--------|
| 21 | -]/[- | LDI | hole selector switch bit 0 I/9 | 0 | |
| 22 | -] [- | AND | hole selector switch bit 1 I/10 | 0 | |
| 152 | | SQO | FILE #N50 MASK High Preset Value (counts to next hole) DEST N7 3 hole preset sequencer CTRL R4 LEN 5 POS 0000H | FFFFH | |
| 42 | -(U)- | RST | force the sequencer to increment on next scan R4/EN | 1 | |

^① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.

File 4, Rung 2

Is identical to the previous rung except that it is only active when the "hole selector switch" is in the "5-hole" position.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|------------------|---|---------|-----------------------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -] [- | LD | hole selector switch bit 0 | I/9 | 0 | |
| 23 | -]/[- | ANI ^② | hole selector switch bit 1 | I/10 | 0 | |
| 152 | | SQO | FILE #N55 | | FFFFH | |
| | | | MASK | | | |
| | | | High Preset Value | | (counts to next hole) | |
| | | | DEST N7 | | | |
| | | | 5 hole preset sequencer | | | |
| | | | CTRL R5 | | | |
| | | | LEN | | 7 | |
| | | | POS | | 0000H | |
| 42 | -(U)- | RST | force the sequencer to increment on the next scan | R5/EN | 1 | |

File 4, Rung 3^③

Is identical to the two previous rungs except that it is only active when the "hole selector switch" is in the "7-hole" position.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|---|---------|-----------------------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -] [- | LD | hole selector switch bit 0 | I/9 | 0 | |
| 22 | -] [- | AND | hole selector switch bit 1 | I/10 | 0 | |
| 152 | | SQO | FILE #N62 | | FFFFH | |
| | | | MASK | | | |
| | | | High Preset Value | | (counts to next hole) | |
| | | | DEST N7 | | | |
| | | | 7 hole preset sequencer | | | |
| | | | CTRL R6 | | | |
| | | | LEN | | 9 | |
| | | | POS | | 0000H | |
| 42 | -(U)- | RST | force the sequencer to increment on the next scan | R6/EN | 1 | |

^① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.

^② This instruction accesses I/O only available with 32 I/O controllers. Therefore, do not include this instruction if you are using a 16 I/O controller.

^③ More rungs will be added to this subroutine at the end of chapter 14.

Using High-Speed Counter Instructions

This chapter contains general information about the high-speed counter instructions and explains how they function in your application program. Each of the instructions includes information on:

- what the instruction symbol looks like
- typical execution time for the instruction
- how to use the instruction
- how to enter the instruction
- what happens to the HSC when going to the RRUN mode

In addition, the last section contains an application example for a paper drilling machine that shows the high-speed counter instructions in use.

High-Speed Counter Instructions

| Mnemonic | Function Code | Name | Purpose | Page |
|----------|---------------|---|--|-------|
| HSC | 170 | High-Speed Counter | Applies configuration to the high-speed counter hardware, updates the image accumulator, enables counting when the HSC rung is true, and disables counting when the HSC rung is false. | 14-4 |
| HSL | 171 | High-Speed Counter Load | Configures the low and high presets, the output patterns, and mask bit patterns. | 14-15 |
| RES | 7 | High-Speed Counter Reset | Writes a zero to the hardware accumulator and image accumulator. | 14-19 |
| RAC | 172 | High-Speed Counter Reset Accumulator | Writes the value specified to the hardware accumulator and image accumulator. | 14-20 |
| HSE | 173 | High-Speed Counter Interrupt Enable | Enables or disables execution of the high-speed counter interrupt subroutine when a high preset, low preset, overflow, or underflow is reached. | 14-21 |
| HSD | 174 | High-Speed Counter Interrupt Disable | | |
| OUT | 40 | Update High-Speed Counter Image Accumulator | Provides you with real-time access to the hardware accumulator value by updating the image accumulator. | 14-23 |

About the High-Speed Counter Instructions

The high-speed counter instructions used in your program configure, control, and monitor the controllers' hardware counter. The hardware counter's accumulator increments or decrements in response to external input signals. When the high-speed counter is enabled, data table counter C0 is used by the program for monitoring the high-speed counter accumulator and status. The high-speed counter operates *independent* to the controller scan.

Since these are *output* instructions, they do not have LD, AND, and OR equivalents.

When using the high-speed counter, make sure you adjust your input filters accordingly. See page 18-12 for more information on input filters.

Before you learn about these instructions, read the overview that follows on the next page. Refer to page 2-23 for information on wiring your controller for high-speed counter applications.

High-Speed Counter Instructions Overview

Use the high-speed counter instructions to perform specific actions after a preset count is reached. These actions include the automatic and immediate execution of the high-speed counter interrupt routine (file 4) and the immediate update of outputs based on a source and mask pattern you set.

Counter Data File Elements

The high-speed counter instructions reference counter C0. The HSC instruction is fixed at C0. It is comprised of three words. Word 0 is the status word, containing 15 status bits. Word 1 is the preset value. Word 2 is the accumulated value. Once assigned to the HSC instruction, C0 is not available as an address for any other counter instructions.

| | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | |
|--------|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---------------|
| Word 0 | CU | CD | DN | OV | UN | UA | HP | LP | IV | IN | IH | IL | PE | LS | IE | | ← Status Word |
| Word 1 | Preset Value | | | | | | | | | | | | | | | | |
| Word 2 | Accumulator Value | | | | | | | | | | | | | | | | |

CU = Counter Up Enable Bit
 CD = Counter Down Enable Bit
 DN = High Preset Reached Bit
 OV = Overflow Occurred Bit
 UN = Underflow Occurred Bit
 UA = Update High-Speed Counter Accumulator Bit
 HP = Accumulator \geq High Preset Bit
 LP = Accumulator \leq Low Preset Bit
 IV = Overflow Caused High-Speed Counter Interrupt Bit
 IN = Underflow Caused High-Speed Counter Interrupt Bit
 IH = High Preset Reached Caused Interrupt Bit
 IL = Low Preset Reached Caused Interrupt Bit
 PE = High-Speed Counter Interrupt Pending Bit
 LS = High-Speed Counter Interrupt Lost Bit
 IE = High-Speed Counter Interrupt Enable Bit

Counter preset and accumulated values are stored as signed integers.

Using Status Bits

The high-speed counter status bits are retentive. When the high-speed counter is first configured, bits 3–7, 14, and 15 are reset and bit 1 (IE) is set.

- **Counter Up Enable Bit CU** (bit 15) is used with all of the high-speed counter types. If the HSC instruction is true, the CU bit is set to one. If the HSC instruction is false, the CU bit is set to zero. Do not write to this bit.
- **Counter Down Enable Bit CD** (bit 14) is used with the Bidirectional Counters (modes 3–8). If the HSC instruction is true, the CD bit is set to one. If the HSC instruction is false, the CD bit is set to zero. Do not write to this bit.
- **High Preset Reached Bit DN** (bit 13) For the Up Counters (modes 1 and 2), this bit is an edge triggered latch bit. This bit is set when the high preset is reached. You can reset this bit with an OTU instruction or by executing an RAC or RES instruction.

The DN bit is a reserved bit for all other Counter options (modes 3–8).

- **Overflow Occurred Bit OV** (bit 12) For the Up Counters (modes 1 and 2), this bit is set by the controller when the high preset is reached if the DN bit is set.

Tip

For the Bidirectional Counters (modes 3–8), the OV bit is set by the controller after the hardware accumulator transitions from 32,767 to –32,768. You can reset this bit with an OTU instruction or by executing an RAC or RES instruction for both the up and bidirectional counters.

- **Underflow Occurred Bit UN** (bit 11) is a reserved bit for the Up Counters (modes 1 and 2). Do not write to this bit.

Tip

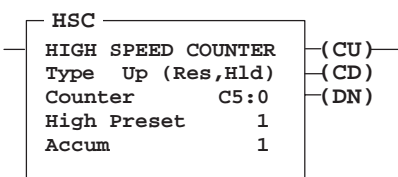
For the Bidirectional Counters (modes 3–8), the UN bit is set by the controller when the hardware accumulator transitions from –32,768 to +32,767. You can reset this bit with an OTU instruction or by executing an RAC or RES instruction.

- **Update High-Speed Counter Accumulator Bit UA** (bit 10) is used with an OTE instruction to update the instruction image accumulator value with the hardware accumulator value. (The HSC instruction also performs this operation each time the rung with the HSC instruction is evaluated as true.)
- **Accumulator \geq High Preset Bit HP** (bit 9) is a reserved bit for all Up Counters (modes 1 and 2).
For the Bidirectional Counters (modes 3–8), if the hardware accumulator becomes greater than or equal to the high preset, the HP bit is set. If the hardware accumulator becomes less than the high preset, the HP bit is reset by the controller. Do not write to this bit. (Exception – you can set or reset this bit during the initial configuration of the HSC instruction. See page 14–23 for more information.)
- **Accumulator \leq Low Preset Bit LP** (bit 8) is a reserved bit for all Up Counters.
For the Bidirectional Counters, if the hardware accumulator becomes less than or equal to the low preset, the LP bit is set by the controller. If the hardware accumulator becomes greater than the low preset, the LP bit is reset by the controller. Do not write to this bit. (Exception – you can set or reset this bit during the initial configuration of the HSC instruction. See page 14–23 for more information.)
- **Overflow Caused High-Speed Counter Interrupt Bit IV** (bit 7) is set to identify an overflow as the cause for the execution of the high-speed counter interrupt routine. The IN, IH, and IL bits are reset by the controller when the IV bit is set. Examine this bit at the start of the high-speed counter interrupt routine (file 4) to determine why the interrupt occurred.
- **Underflow caused User Interrupt Bit IN** (bit 6) is set to identify an underflow as the cause for the execution of the high-speed counter interrupt routine. The IV, IH, and IL bits are reset by the controller when the IN bit is set. Examine this bit at the start of the high-speed counter interrupt routine (file 4) to determine why the interrupt occurred.

- **High Preset Reached Caused User Interrupt Bit IH** (bit 5) is set to identify a high preset reached as the cause for the execution of the high-speed counter interrupt routine. The IV, IN, and IL bits are reset by the controller when the IH bit is set. Examine this bit at the start of the high-speed counter interrupt routine (file 4) to determine why the interrupt occurred.
- **Low Preset Reached Caused High-Speed Counter Interrupt Bit IL** (bit 4) is set to identify a low preset reached as the cause for the execution of the high-speed counter interrupt routine. The IV, IN, and IH bits are reset by the controller when the IL bit is set. Examine this bit at the start of the high-speed counter interrupt routine (file 4) to determine why the interrupt occurred.
- **High-Speed Counter Interrupt Pending Bit PE** (bit 3) is set to indicate that a high-speed counter interrupt is waiting for execution. This bit is cleared by the controller when the high-speed counter interrupt routine begins executing. This bit is reset if an RAC or RES instruction is executed. Do not write to this bit.
- **High-Speed Counter Interrupt Lost Bit LS** (bit 2) is set if an high-speed counter interrupt occurs while the PE bit is set. You can reset this bit with an OTU instruction or by executing an RAC or RES instruction.
- **High-Speed Counter Interrupt Enable Bit IE** (bit 1) is set when the high-speed counter interrupt is enabled to run when an high-speed counter interrupt condition occurs. It is reset when the interrupt is disabled. This bit is also set when the high-speed counter is first configured. Do not write to this bit.

High-Speed Counter (HSC)

Ladder representation:



Execution Times (μ sec) when:

| True | False |
|-------|-------|
| 21.00 | 21.00 |

Use this instruction to configure the high-speed counter. Only one HSC instruction can be used in a program. The high-speed counter is not operational until the first true execution of the HSC instruction. When the HSC rung is false, the high-speed counter is disabled from *counting* but all other HSC features are operational.

The Counter address of the HSC instruction is fixed at C0.

After the HSC is configured, the image accumulator is updated with the current hardware accumulator value every time the HSC instruction is evaluated as true or false.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Type** indicates the counter selected. Refer to page 14–5 for making your high-speed counter selection. Each type is available with reset and hold functionality.
- **High Preset** is the accumulated value that triggers a user-specified action such as updating outputs or generating an high-speed counter interrupt.
- **Accumulator** is the number of accumulated counts.

The table that follows uses the terminology shown here to indicate the status of counting:

- Up↑ – increments by 1 when the input energizes (edge).
- Down↑ – decrements by 1 when the input energizes (edge).
- Reset↑ – resets the accumulator to zero when the input energizes (edge).
- Hold – disables the high-speed counter from counting while the input is energized (level).
- Count – increments or decrements by 1 when the input energizes (edge).
- Direction – allows up counts when the input is de-energized and down counts while the input is energized (level).
- A – input pulse in an incremental (quadrature) encoder (edge/level)
- B – input pulse in an incremental (quadrature) encoder (edge/level)
- Z – reset pulse in an incremental (quadrature) encoder (edge/level)
- ↑ – the signal is active on the rising edge only (off to on).

The table below lists the types of high-speed counter you can choose.

| High-Speed Counter Type | High-Speed Counter Functionality | Input Terminal Used | | | | Page |
|---|---|---------------------|-----------|----------|----------|-------|
| | | I/0 | I/1 | I/2 | I/3 | |
| Up | Up Counter operation uses a single-ended input. | Up↑ | Not Used | Not Used | Not Used | 14-5 |
| Up (with external reset and hold) | Up Counter operation uses a single-ended input with external reset and hold inputs. | Up↑ | Not Used | Reset↑ | Hold | |
| Pulse and direction | Bidirectional operation uses both pulse and direction inputs. | Count↑ | Direction | Not Used | Not Used | 14-9 |
| Pulse and direction (with external reset and hold) | Bidirectional operation uses both pulse and direction inputs with external reset and hold inputs. | Count↑ | Direction | Reset↑ | Hold | |
| Up and down | Bidirectional operation uses both up and down inputs. | Up↑ | Down↑ | Not Used | Not Used | 14-10 |
| Up and down (with external reset and hold) | Bidirectional operation uses both up and down pulse inputs with external reset and hold inputs. | Up↑ | Down↑ | Reset↑ | Hold | |
| Encoder | Bidirectional operation uses quadrature encoder inputs. | A | B | Not Used | Not Used | 14-12 |
| Encoder (with external reset and hold) | Bidirectional operation uses both quadrature encoder inputs with external reset and hold inputs. | A | B | Z | Hold | |

One difference between Up Counters and Bidirectional Counters is that for Bidirectional Counters the accumulator and preset values are not changed by the high-speed counter when the presets are reached. The RAC and HSL instructions must be used for this function. The Up Counters clear the accumulator values and re-load the high preset values when the previous preset is reached.

Entering the Instruction

You enter the instruction from within the program monitor functional area. The following items apply when entering the instruction:

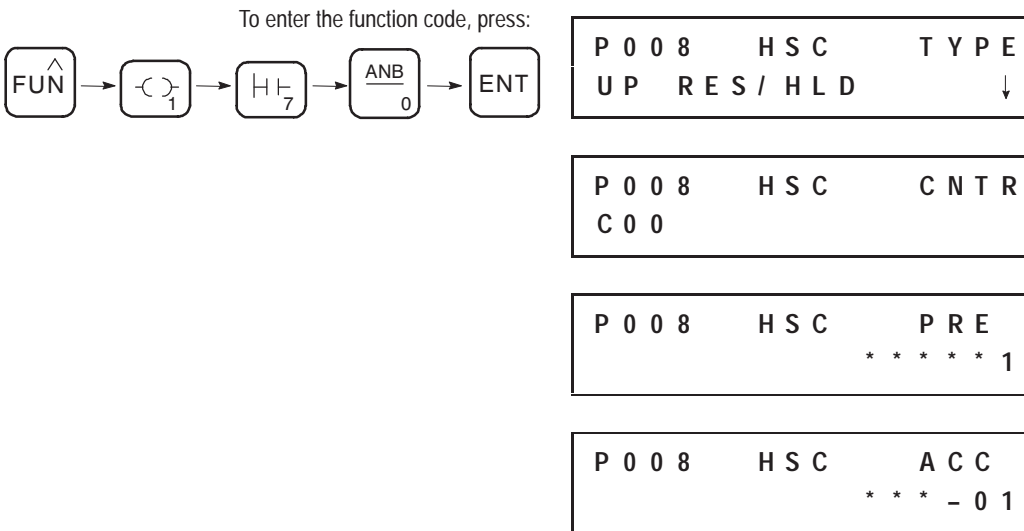
- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- If you see a down arrow on the display it means there are more options available. To scroll through the options, press this key:



- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17-4.)



Once instruction entry is complete, the parameters are condensed to two screens as shown here:

| | | |
|---------|---------------|---------|
| P 0 0 8 | H S C | T Y P E |
| U P | R E S / H L D | |

| | | |
|---------|-------|-------|
| P 0 0 8 | H S C | C 0 0 |
| P | 1 A | - 1 |

Using the Up Counter and the Up Counter with Reset and Hold

Up counters are used when the parameter being measured is uni-directional, such as material being fed into a machine or as a tachometer recording the number of pulses over a given time period.

Both types of Up Counters operate identically, except that the Up Counter with reset and hold uses external inputs 2 and 3.

For the Up Counter, each Off-to-On state change of input I0/0 adds 1 to the accumulator until the high preset is reached. The accumulator is then automatically reset to zero. The Up Counter operates in the 0 to +32,767 range inclusive and can be reset to zero using the Reset (RES) instruction.

When the HSC instruction is first executed true, the:

- Accumulator C0.ACC is loaded to the hardware accumulator.
- High preset C0.PRE is loaded to the hardware high preset.

Operation

Tip

If you move data to the high preset without using the RAC instruction (with a MOV) after the high-speed counter is configured, the data is loaded to the instruction image but is *not* loaded to the hardware. The modified high preset value is not loaded to the hardware until the existing hardware high preset is reached or an RAC or RES instruction is executed.

The high preset value loaded to the hardware must be between 1 and 32,767 inclusive or an error (37H) occurs. Any value between -32,768 and +32,767 inclusive can be loaded to the hardware accumulator.

| The Following Condition | Occurs when |
|--------------------------|--|
| A high preset is reached | either the hardware accumulator transitions from the hardware high preset -1 to the hardware high preset, or |
| | the hardware accumulator is loaded with a value greater than or equal to the hardware high preset, or |
| | the hardware high preset is loaded with a value that is less than or equal to the hardware accumulator. |

When a high preset is reached, no counts are lost.

- Hardware and instruction accumulators are reset.
- Instruction high preset is loaded to the hardware high preset.
- If the DN bit is not set, the DN bit is set. The IH bit is also set and the IL, IV, and IN bits are reset.
- If the DN bit is already set, the OV bit is set. The IV bit is also set and the IL, IV and IN bits are reset.
- High-speed counter interrupt file (file 4) is executed if the interrupt is enabled.

The following tables summarize what the input state must be for the corresponding high-speed counter action to occur:

Up Counter

| Input State | | | | | High-Speed Counter Action |
|-------------------|-----------------------|-------------------|------------------|----------|---------------------------|
| Input Count (I/O) | Input Direction (I/1) | Input Reset (I/2) | Input Hold (I/3) | HSC Rung | |
| Turning Off-to-On | NA | NA | NA | True | Count Up |
| NA | NA | NA | NA | False | Hold Count |

NA (Not Applicable)

Up Counter with Reset and Hold

| Input state | | | | | High-Speed Counter Action |
|-------------------------|-----------------------|-------------------------|------------------|----------|---------------------------|
| Input Count (I/O) | Input Direction (I/1) | Input Reset (I/2) | Input Hold (I/3) | HSC Rung | |
| Turning Off-to-On | NA | Off, On, or Turning Off | Off | True | Count Up |
| NA | NA | Off, On, or Turning Off | On | NA | Hold Count |
| NA | NA | Off, On, or Turning Off | NA | False | Hold Count |
| Off, On, or Turning Off | NA | Off, On, or Turning Off | NA | NA | Hold Count |
| NA | NA | Turning On | NA | NA | Reset to 0 |

NA (Not Applicable)

Using the Bidirectional Counter and the Bidirectional Counter with Reset and Hold

Bidirectional counters are used when the parameter being measured can either increment or decrement. For example, a package entering and leaving a storage bin is counted to regulate flow through the area.

The Bidirectional Counters operate identically except for the operation of inputs 1 and 0. For the Pulse and Direction type, input 0 provides the pulse and input 1 provides the direction. For the Up and Down type, input 0 provides the Up count and input 1 provides the Down count. Both types are available with and without reset and hold. Refer to page 14–5 for more information regarding Bidirectional Counter types.

For the Bidirectional Counters, both high and low presets are used. The low preset value must be less than the high preset value or an error (37H) occurs. The hardware low preset default is $-32,768$.

Bidirectional Counters operate in the $-32,768$ to $+32,767$ range inclusive and can be reset to zero using the Reset (RES) instruction.

Operation

When the HSC instruction is first executed true, the:

- Instruction accumulator is loaded to the hardware accumulator.
- Instruction high preset is loaded to the hardware high preset.

After the first true HSC instruction execution, data can only be transferred to the hardware accumulator via the RES or RAC instruction or to the hardware high and low presets via the HSL instruction.

Any instruction accumulator value between $-32,768$ and $+32,767$ inclusive can be loaded to the hardware.

| The Following Condition | Occurs when |
|--------------------------|--|
| A high preset is reached | either the hardware accumulator transitions from the hardware high preset -1 to the hardware high preset, or |
| | the hardware accumulator is loaded with a value greater than or equal to the hardware high preset, or |
| | the hardware high preset is loaded with a value that is less than or equal to the hardware accumulator. |

When a high preset is reached, the:

- HP bit is set.
- High-speed counter interrupt file (file 4) is executed if the interrupt is enabled. The IH bit is set and the IL, IV, and IN bits are reset.

Unlike the Up Counters, the accumulator value is not reset and the high preset value is not loaded from the image to the hardware high preset register.

| The Following Condition | Occurs when |
|-------------------------|--|
| A low preset is reached | either the hardware accumulator transitions from the hardware low preset +1 to the hardware low preset, or |
| | the hardware accumulator is loaded with a value less than or equal to the hardware low preset, or |
| | the hardware low preset is loaded with a value that is greater than or equal to the hardware accumulator. |

When the low preset is reached, the:

- LP bit is set.
- High-speed counter interrupt file (file 4) is executed if the interrupt is enabled. The IL bit is set and the IH, IV, and IN bits are reset.

An overflow occurs when the hardware accumulator transitions from +32,767 to -32,768. When an overflow occurs, the:

- OV bit is set.
- High-speed counter interrupt file (file 4) is executed if the interrupt is enabled. The IV bit is set and the IH, IL, and IN bits are reset.

An underflow occurs when the hardware accumulator transitions from -32,768 to +32,767. When an underflow occurs, the:

- UN bit is set.
- High-speed counter interrupt file (file 4) is executed if the interrupt is enabled. The IN bit is set and the IH, IL, and IV bits are reset.

The following tables summarize what the input state must be for the corresponding high-speed counter action to occur:

Bidirectional Counter (Pulse/direction)

| Input Count (I/0) | Input State | | | | High-Speed Counter Action |
|-------------------|-----------------------|-------------------|------------------|----------|---------------------------|
| | Input Direction (I/1) | Input Reset (I/2) | Input Hold (I/3) | HSC Rung | |
| Turning Off-to-On | Off | NA | NA | True | Count Up |
| Turning Off-to-On | On | NA | NA | True | Count Down |
| NA | NA | NA | NA | False | Hold Count |

NA (Not Applicable)

Bidirectional Counter with Reset and Hold (Pulse/direction)

| Input State | | | | | High-Speed Counter Action |
|-------------------------|-----------------------|-------------------------|------------------|----------|---------------------------|
| Input Count (I/0) | Input Direction (I/1) | Input Reset (I/2) | Input Hold (I/3) | HSC Rung | |
| Turning Off-to-On | Off | Off, On, or Turning Off | Off | True | Count Up |
| Turning Off-to-On | On | Off, On, or Turning Off | Off | True | Count Down |
| NA | NA | Off, On, or Turning Off | NA | False | Hold Count |
| NA | NA | Off, On, or Turning Off | On | NA | Hold Count |
| Off, On, or Turning Off | NA | Off, On, or Turning Off | NA | NA | Hold Count |
| NA | NA | Turning On | NA | NA | Reset to 0 |

NA (Not Applicable)

Bidirectional Counter (Up/down count)

| Input State | | | High-Speed Counter Action |
|-------------------------|-------------------------|----------|---------------------------|
| Input Up Count (I/0) | Input Down Count (I/1) | HSC Rung | |
| Turning Off-to-On | Off, On, or Turning Off | True | Count Up |
| Off, On, or Turning Off | Turning Off-to-On | True | Count Down |
| NA | NA | False | Hold Count |

NA (Not Applicable)

Bidirectional Counter with Reset and Hold (Up/down count)

| Input State | | | | | High-Speed Counter Action |
|-------------------------|-------------------------|-------------------------|------------------|----------|---------------------------|
| Input Up Count (I/0) | Input Down Count (I/1) | Input Reset (I/2) | Input Hold (I/3) | HSC Rung | |
| Turning Off-to-On | Off, On, or Turning Off | Off, On, or Turning Off | Off | True | Count Up |
| Off, On, or Turning Off | Turning Off-to-On | Off, On, or Turning Off | Off | True | Count Down |
| NA | NA | Off, On, or Turning Off | NA | False | Hold Count |
| NA | NA | Off, On, or Turning Off | On | NA | Hold Count |
| Off, On, or Turning Off | Off, On, or Turning Off | Off, On, or Turning Off | NA | NA | Hold Count |
| NA | NA | Turning On | NA | NA | Reset to 0 |

NA (Not Applicable)

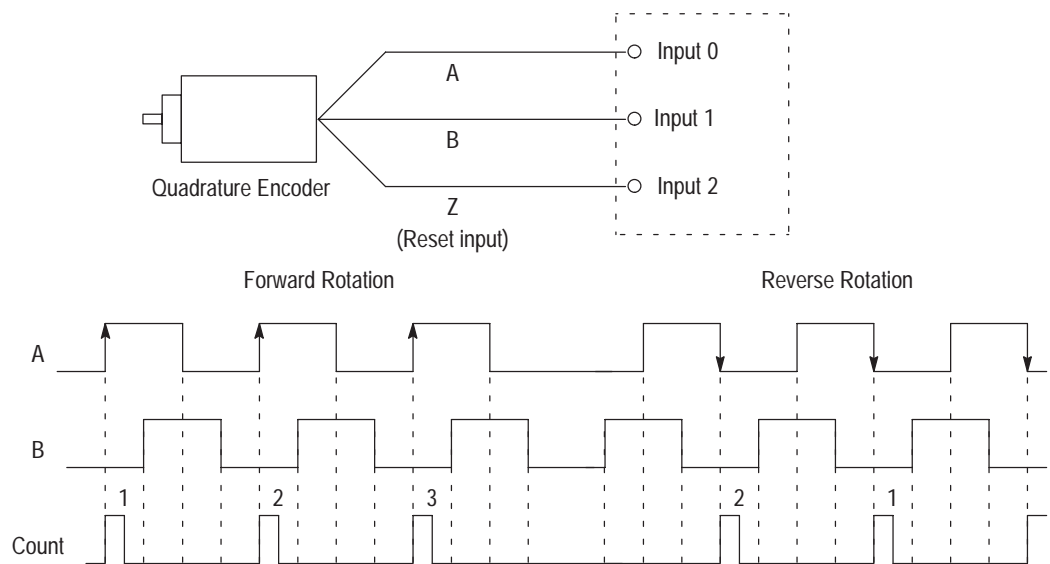
When up and down input pulses occur simultaneously, the high-speed counter counts up, then down.

Using the Bidirectional Counter with Reset and Hold with a Quadrature Encoder

The Quadrature Encoder is used for determining direction of rotation and position for rotating, such as a lathe. The Bidirectional Counter counts the rotation of the Quadrature Encoder.

Bidirectional Counters operate in the $-32,768$ to $+32,767$ range inclusive and can be reset to zero using the reset (RES) instruction. The figure below shows a quadrature encoder connected to inputs 0, 1, and 2. The count direction is determined by the phase angle between A and B. If A leads B, the counter increments. If B leads A, the counter decrements.

The counter can be reset using the Z input. The Z outputs from the encoders typically provide one pulse per revolution.



Operation

For the Bidirectional Counters, both high and low presets are used. The low preset value must be less than the high preset value or an error (37H) occurs.

When the HSC instruction is first executed true, the:

- Instruction accumulator is loaded to the hardware accumulator.
- Instruction high preset is loaded to the hardware high preset.

Any instruction accumulator value between $-32,768$ and $+32,767$ inclusive can be loaded to the hardware.

After the first true HSC instruction execution, data can only be transferred to the hardware accumulator via an RES or RAC instruction, or to the hardware high and low presets via the HSL instruction.

| The Following Condition | Occurs when |
|--------------------------|--|
| A high preset is reached | either the hardware accumulator transitions from the hardware high preset -1 to the hardware high preset, or |
| | the hardware accumulator is loaded with a value greater than or equal to the hardware high preset, or |
| | the hardware high preset is loaded with a value that is less than or equal to the hardware accumulator. |

When a high preset is reached, the:

- HP bit is set.
- High-speed counter interrupt file (file 4) is executed if the interrupt is enabled. The IH bit is set and the IL, IN, and IV bits are reset.

Unlike the Up Counters, the accumulator value does not reset and the high preset value does not get loaded from the image to the hardware high preset register.

| The Following Condition | Occurs when |
|-------------------------|--|
| A low preset is reached | either the hardware accumulator transitions from the hardware low preset $+1$ to the hardware low preset, or |
| | the hardware accumulator is loaded with a value less than or equal to the hardware low preset, or |
| | the hardware low preset is loaded with a value that is greater than or equal to the hardware accumulator. |

When a low preset is reached, the:

- LP bit is set.
- High-speed counter interrupt file (file 4) is executed if the interrupt is enabled. The IL bit is set and the IH, IN, and IV bits are reset.

An overflow occurs when the hardware accumulator transitions from +32,767 to -32,768. When an overflow occurs, the:

- OV bit is set.
- High-speed counter interrupt file (file 4) is executed if the interrupt is enabled. The IV bit is set and the IH, IL, and IN bits are reset.

An underflow occurs when the hardware accumulator transitions from -32,768 to +32,767. When an underflow occurs, the:

- UN bit is set.
- High-speed counter interrupt file (file 4) is executed if the interrupt is enabled. The IN bit is set and the IH, IL, and IV bits are reset.

The following tables summarize what the input state must be for the corresponding high-speed counter action to occur:

Bidirectional Counter (Encoder)

| Input State | | | High-Speed Counter Action |
|---------------|---------------|----------|---------------------------|
| Input A (I/0) | Input B (I/1) | HSC Rung | |
| Turning On | Off | True | Count Up |
| Turning Off | Off | True | Count Down |
| NA | On | NA | Hold Count |
| NA | NA | False | Hold Count |

NA (Not Applicable)

Bidirectional Counter with Reset and Hold (Encoder)

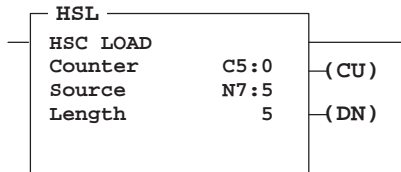
| Input State | | | | | High-Speed Counter Action |
|---------------|---------------|-----------------|------------------|----------|---------------------------|
| Input A (I/0) | Input B (I/1) | Input Z (I/2) | Input Hold (I/3) | HSC Rung | |
| Turning On | Off | Off | Off | True | Count Up |
| Turning Off | Off | Off | Off | True | Count Down |
| Off or On | NA | Off | NA | NA | Hold Count |
| NA | On | Off | NA | NA | Hold Count |
| NA | NA | Off | NA | False | Hold Count |
| NA | NA | Off | On | NA | Hold Count |
| Off | Off | On ^① | NA | NA | Reset to 0 |

NA (Not Applicable)

① The optional hardware high-speed counter reset is the logical coincidence of $\bar{A} \times \bar{B} \times Z$.

High-Speed Counter Load (HSL)

Ladder representation:



Execution Times (μ sec) when:

| True | False |
|-------|-------|
| 66.00 | 7.00 |

This instruction allows you to set the low and high presets, low and high output source, and the output mask. When either a high or low preset is reached, you can instantly update selected outputs.

If you are using the HSL instruction with the Up Counter, the high preset must be ≥ 1 and $\leq +32,767$ or an error (37H) occurs. For the bidirectional counters, the high preset must be greater than the low preset or an error (37H) occurs.

The Counter referenced by this instruction has the same address as the HSC instruction counter and is fixed at C0.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** is an address that identifies the first of five data words used by the HSL. The source can be either an integer or binary file element.
- **Length** is the number of elements starting from the source. This number is always 5.

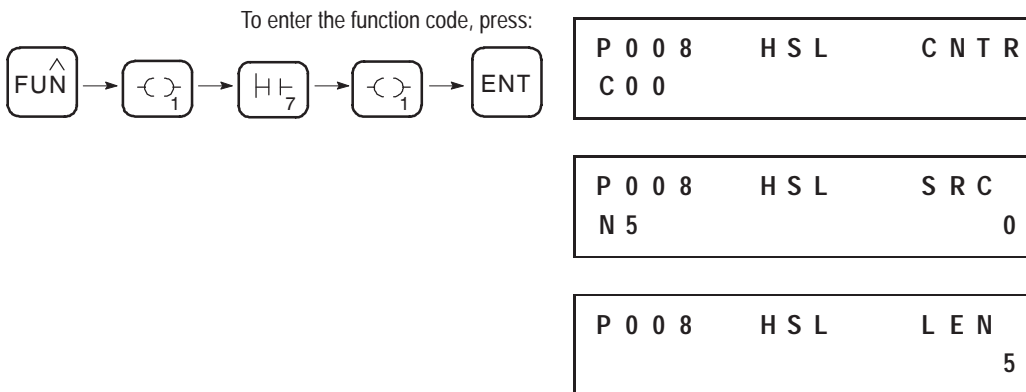
Entering the Instruction

You enter the instruction from within the program monitor functional area. The following items apply when entering the instruction:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17–4.)



Operation

The HSL instruction allows you to *configure* the high-speed counter to instantaneously and automatically update external outputs whenever a high or low preset is reached. The physical outputs are automatically updated in less than 30 μ s. (The physical turn-on time of the outputs is *not* included in this amount.) The output image is then automatically updated at the next poll for user interrupts or at the next IOM instruction or output scan, whichever occurs first.

With this instruction, you can change the high preset for the up counters or both the high and low presets for Bidirectional Counters during run. You can also modify the output mask configuration during run.

The source address is either an integer or binary file element. For example, if N5 is selected as the source address, the additional parameters for the execution of this instruction would appear as shown on the following page.

| Parameter Image Location | Up Counter Only | Bidirectional Counters | Description |
|--------------------------|-----------------|------------------------|--|
| N5 | Output Mask | Output Mask | Identifies which group of four bits in the output file (word 0) are controlled. 000F=bits 3-0 00F0=bits 7-4 0003=bits 0 and 1 00FF= bits 7-0 |
| N6 | Output Source | Output High Source | (Up count) The status of bits in this word are written "through" the mask to the actual outputs. |
| N7 | High Preset | High Preset | (Up count) When the accumulator reaches this value, the output source is written through the output mask to the actual outputs and the HSC subroutine (file 4) is scanned. |
| N8 | Reserved | Output Low Source | (Down count) The status of bits in this word are written "through" the mask to the actual outputs. |
| N9 | Reserved | Low Preset | (Down count) When the accumulator reaches this value, the output source is written through the output mask to the actual outputs and the HSC subroutine (file 4) is scanned. |

The bits in the output mask directly correspond to the physical outputs. If a bit is set to 1, the corresponding output can be changed by the high-speed counter. If a bit is set to 0, the corresponding output cannot be changed by the high-speed counter.

The bits in the high and low sources also directly correspond to the physical outputs. The high source is applied when the high preset is reached. The low source is applied when the low preset is reached. The final output states are determined by applying the output source over the mask and updating only the unmasked outputs (those with a 1 in the mask bit pattern).

You can always change the state of the outputs via the user program or programming device regardless of the output mask. The high-speed counter only modifies selected outputs and output image bits based on source and mask bit patterns when the presets are reached. The last device that changes the output image (i.e., user program or high-speed counter) determines the actual output pattern.



ATTENTION: Forces override any output control from either the high-speed counter or from the output image. Forces may also be applied to the high-speed counter inputs. Forced inputs are recognized by the high-speed counter (e.g. a forced count input off and on increments the high-speed accumulator).

The high-speed counter hardware is updated immediately when the HSL instruction is executed regardless of high-speed counter type (Up Counter or Bidirectional Counter). For the Up Counters, the last two registers are ignored since the low preset does not apply.

If a fault occurs due to the HSL instruction, the HSL parameters are not loaded to the high-speed counter hardware. You can use more than one HSL instruction in your program. The HSL instructions can have different image locations for the additional parameters.

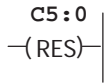


ATTENTION: Do not change a preset value *and* an output mask/source with the same HSL instruction as the accumulator is approaching the old preset value.

If the high-speed counter is enabled and the HSL instruction is evaluated true, the high-speed counter parameters in the HSL instruction are applied immediately without stopping the operation of the high-speed counter. If the same HSL instruction is being used to change the high-speed counter controlled mask/source and the preset, the mask/source is changed first and the preset second. (The preset is changed within 40 μ s after the mask/source.) If the original preset is reached after the new mask/source is applied but before the new preset is applied, the new outputs are applied immediately.

High-Speed Counter Reset (RES)

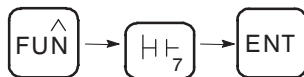
Ladder representation:



Execution Times (μsec) when:

| True | False |
|-------|-------|
| 51.00 | 6.00 |

To enter the function code, press:



The RES instruction allows you to write a zero to the hardware accumulator and image accumulator.

The Counter referenced by this instruction has the same address as the HSC instruction counter and is entered as C0.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

```
P 0 0 0   R E S
C 0 0
```

Operation

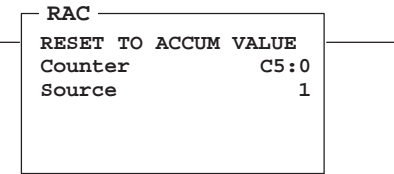
Execution of this instruction immediately:

- removes pending high-speed counter interrupts
- resets the hardware and instruction accumulators
- resets the PE, LS, OV, UN, and DN status bits
- loads the instruction high preset to the hardware high preset (if the high-speed counter is configured as an up counter)
- resets the IL, IH, IN, or IV status bits

You can have more than one RES instruction in your program.

High-Speed Counter Reset Accumulator (RAC)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|-------|-------|
| 56.00 | 6.00 |

This instruction allows you to write a specific value to the hardware accumulator and image accumulator.

The Counter referenced by this instruction has the same address as the HSC instruction counter and is fixed at C0.

Entering Parameters

Enter the following parameter when programming this instruction:

- **Source** represents the value that is loaded to the accumulator. The source can be a constant or an address.

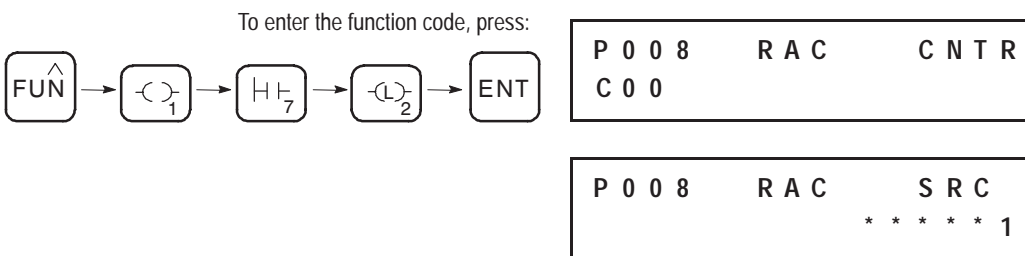
Entering the Instruction

You enter the instruction from within the program monitor functional area. The following items apply when entering the instruction:

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17-4.)



Once instruction entry is complete, the parameters appear as shown here:

| | | |
|---------|-------|-------|
| P 0 0 8 | R A C | C 0 0 |
| P | 1 A | - 1 |

| | | |
|---------|-------|-------|
| P 0 0 8 | R A C | S R C |
| | | 1 |

Operation

Execution of the RAC:

- removes pending high-speed counter interrupts
- resets the PE, LS, OV, UN, and DN status bits
- loads a new accumulator value to the hardware and instruction image
- loads the instruction high preset to the hardware high preset (if the high-speed counter is configured as an Up Counter)
- resets the IL, IH, IN, or IV status bits

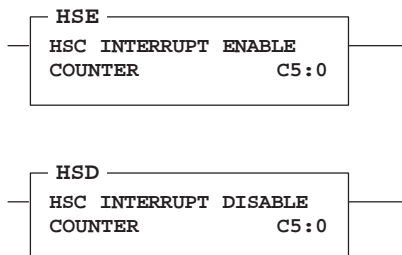
The source can be a constant or any integer element in files 0–7. The hardware and instruction accumulators are updated with the new accumulator value immediately upon instruction execution.

You can have more than one RAC instruction per program referencing the same source or different sources.

High-Speed Counter Interrupt Enable (HSE) and Disable (HSD)

These instructions enable or disable a high-speed counter interrupt when a high preset, low preset, overflow, or underflow is reached. Use the HSD and HSE in pairs to provide accurate execution for your application.

Ladder representation:



The Counter referenced by these instructions has the same address as the HSC instruction counter and is fixed at C0.

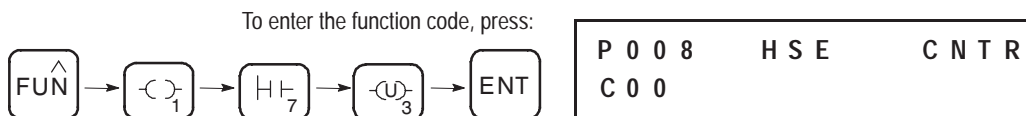
Execution Times (μsec) when:

| | True | False |
|-----|-------|-------|
| HSE | 10.00 | 7.00 |
| HSD | 8.00 | 7.00 |

Using HSE

Entering the Instruction

You enter the instruction from within the program monitor functional area.



Operation

When the high-speed counter interrupt is enabled, user subroutine (file 4) is executed when:

- A high or low preset is reached.
- An overflow or underflow occurs.

When in RSSN mode and in an idle condition, the high-speed counter interrupt is held off until the next scan trigger is received from the programming device. The high-speed counter accumulator counts while idle.

If the HSE is subsequently executed after the pending bit is set, the interrupt is executed immediately.

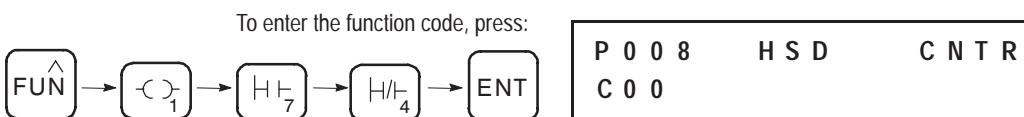
The default state of the high-speed counter interrupt is enabled (the IE bit is set to 1).

If the high-speed counter interrupt routine is executing and another high-speed counter interrupt occurs, the second high-speed counter interrupt is saved but is considered pending. (The PE bit is set.) The second interrupt is executed immediately after the first one is finished executing. If a high-speed counter interrupt occurs while a high-speed counter interrupt is pending, the most recent high-speed counter interrupt is lost and the LS bit is set.

Using HSD

Entering the Instruction

You enter the instruction from within the program monitor functional area.



Operation

The HSD instruction disables the high-speed counter interrupt, preventing the interrupt subroutine from being executed.

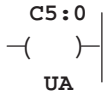
If the HSE is subsequently executed after the pending bit is set, the interrupt is executed immediately.

This HSD instruction does not cancel an interrupt, but results in the pending bit (C0/PE) being set when:

- A high or low preset is reached.
- An overflow or underflow occurs.

Update High-Speed Counter Image Accumulator (OUT)

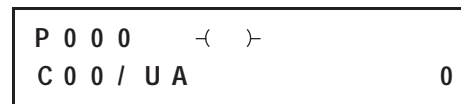
Ladder representation:



Execution Times (μsec) when:

| True | False |
|-------|-------|
| 12.00 | 7.00 |

To enter the OUT instruction, press:



When an OUT bit instruction is addressed for the high-speed counter (C0) UA bit, the value in the hardware accumulator is written to the value in the image accumulator (C0.ACC). This provides you with real-time access to the hardware accumulator value. This is in addition to the automatic transfer from the hardware accumulator to the image accumulator that occurs each time the HSC instruction is evaluated.

Entering the Instruction

You enter the instruction from within the program monitor functional area.

Operation

This instruction transfers the hardware accumulator to the instruction accumulator. When the OUT instruction is executed true, the hardware accumulator is loaded to the instruction image accumulator (C0.ACC).

What Happens to the HSC When Going to RRUN Mode

Once initialized, the HSC instruction retains its previous state when going through a mode change or power cycle. This means that the HSC Accumulator (C0.ACC) and High Preset values are retained. Outputs under the direct control of the HSC also retain their previous state. The Low Preset Reached and High Preset Reached bits (C0/LP and C0/HP) are also retained. They are examined by the HSC instruction during the high-speed counter's first true evaluation in the RRUN mode to differentiate a retentive RRUN mode entry from an external or initial Accumulator (C0.ACC) modification.

At the first true HSC instruction execution after going to run, the Low Preset is initialized to -32,768 and the output mask and high and low output patterns are initialized to zero. Use the HSL instruction during the first pass to restore any values necessary for your application.

You can modify the behavior of the high-speed counter at RRUN mode entry by adjusting the HSC parameters prior to the first true execution of the HSC instruction. The following example ladder rungs and instruction lists demonstrate different ways to adjust the HSC parameters.

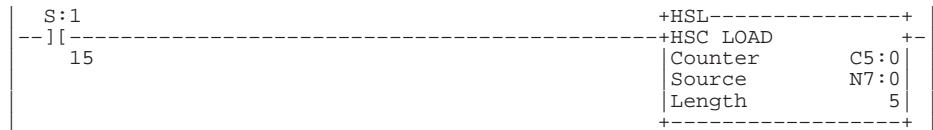
Example 1

To enter the RRUN mode and have the HSC Outputs, ACC, and Interrupt Subroutine resume their previous state, apply the following:

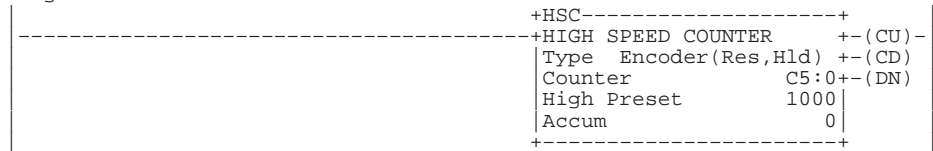
Ladder Rungs

Rung 2:0

No action required. (Remember that all OUT instructions are zeroed when entering the RRUN mode. Use SET/RST instructions in place of OUT instructions in your conditional logic requiring retention.)



Rung 2:1



Instruction List

File 2, Rung 1

No action required. (Remember that all OUT instructions are zeroed when entering the RRUN mode. Use SET/RST instructions in place of OUT instructions in your conditional logic requiring retention.)

| FUN | GRAPHIC | MNEMONIC | PARAMETER | VALUE | FORCES |
|------|---------|----------|--------------------------|-------|--------|
| CODE | SYMBOL | | NAME ADDRESS | | |
| 20 | -] [- | LD | S1/15 | 0 | |
| 171 | | HSL | CNTR C0 SRC N0 LEN | 5 | |

File 2, Rung 1

| FUN | GRAPHIC | MNEMONIC | PARAMETER | VALUE | FORCES |
|------|---------|----------|-------------------------------|------------------------------------|--------|
| CODE | SYMBOL | | NAME ADDRESS | | |
| 170 | | HSC | TYPE CNTR C0 PRE ACC | Encoder (Res,Hld) 1000 0000H | |

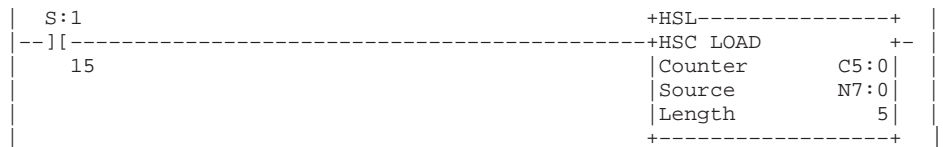
Example 2

To enter the RRUN mode and retain the HSC ACC value while having the HSC Outputs and Interrupt Subroutine reassert themselves, apply the following:

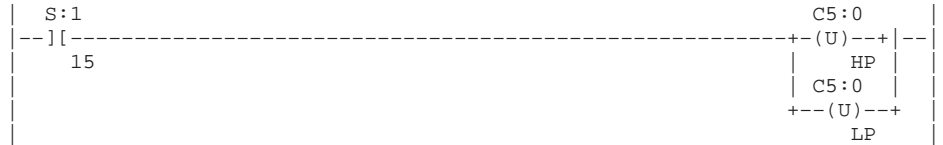
Ladder Rungs

Rung 2:0

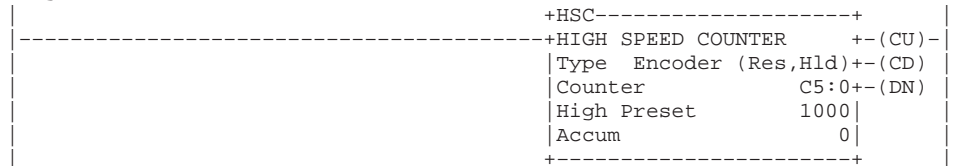
Unlatch the C5:0/HP and C5:0/LP bits during the first scan BEFORE the HSC instruction is executed for the first time.



Rung 2:1



Rung 2:2



Instruction List

File 2, Rung 0

Unlatch the C5:0/HP and C5:0/LP bits during the first scan BEFORE the HSC instruction is executed for the first time.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME | ADDRESS | VALUE | FORCES |
|----------|----------------|----------|--------------------------|---------|-------|--------|
| 20 | -] [- | LD | S1 | 15 | 0 | |
| 171 | | HSL | CNTR C0 SRC N0 LEN | | 5 | |

File 2, Rung 1

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME | ADDRESS | VALUE | FORCES |
|----------|----------------|----------|----------------|---------|-------|--------|
| 20 | -] [- | LD | S1 | 15 | 0 | |
| 42 | - (U) - | RST | C0 | HP | 0 | |
| 42 | - (U) - | RST | C0 | LP | 0 | |

File 2, Rung 2

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME | ADDRESS | VALUE | FORCES |
|----------|----------------|----------|-------------------------------|---------|------------------------------------|--------|
| 170 | | HSC | TYPE CNTR C0 PRE ACC | | Encoder (Res,Hld) 1000 0000H | |

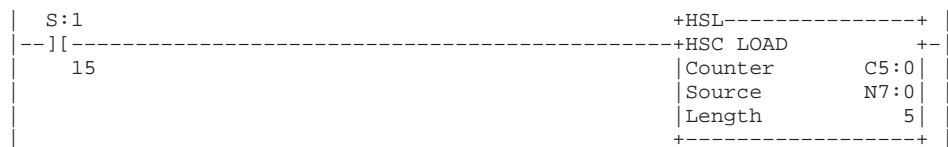
Example 3

To enter the RRUN mode and have the HSC ACC and Interrupt Subroutine resume their previous state, while externally initializing the HSC outputs, apply the following:

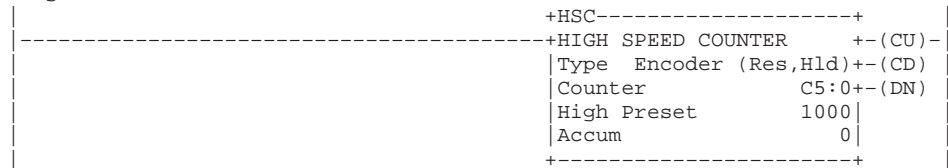
Ladder Rungs

Rung 2:0

Unlatch or latch the output bits under HSC control during the first scan after the HSC instruction is executed for the first time. (Note, you *could* place this rung before the HSC instruction; however, this is not recommended.)

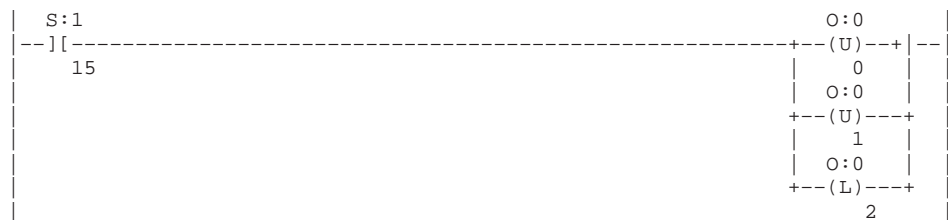


Rung 2:1



Rung 2:2

This rung is programmed with the knowledge of an HSL mask of 0007 (Outputs 0-2 are used) and initializes the HSC outputs each RRUN mode entry. Outputs O/0 and O/1 are off, while Output O/2 is on.



Instruction List

File 2, Rung 0

Unlatch or latch the output bits under HSC control during the first scan after the HSC instruction is executed for the first time. (Note, you could place this rung before the HSC instruction; however, this is not recommended.)

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|-----------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -) [- | LD | | S1/15 | 0 | |
| 171 | | HSL | CNTR | C0 | | |
| | | | SRC | N0 | | |
| | | | LEN | | 5 | |

File 2, Rung 1

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|-----------|---------|-------------------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 170 | | HSC | TYPE | | Encoder (Res,Hld) | |
| | | | CNTR | C0 | | |
| | | | PRE | | 1000 | |
| | | | ACC | | 0000H | |

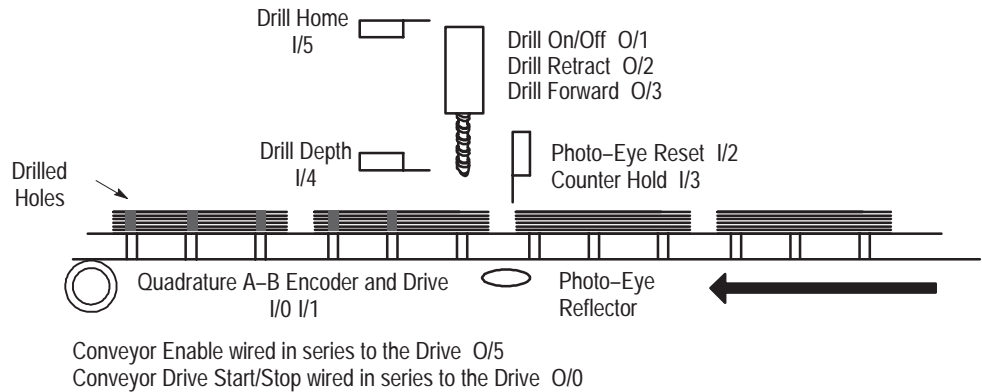
File 2, Rung 2

This rung is programmed with the knowledge of an HSL mask of 0007 (Outputs 0-2 are used) and initializes the HSC outputs each RRUN mode entry. Outputs O/0 and O/1 are off, while Output O/2 is on.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|-----------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -) [- | LD | | S1/15 | 0 | |
| 42 | -(U)- | RST | | O/0 | 0 | |
| 42 | -(U)- | RST | | O/1 | 0 | |
| 41 | -(L)- | SET | | O/2 | 0 | |

High-Speed Counter Instructions in the Paper Drilling Machine Application Example

To demonstrate the use of the HSC instruction, this section provides ladder rungs followed by the optimized instruction list for these rungs. The rungs are part of the paper drilling machine application example started in chapter 5. Refer to appendix E for the complete paper drilling machine application example.



20226

The main program file (file 2) initializes the HSC instruction, monitors the machine start and stop buttons, and calls other subroutines necessary to run the machine. Refer to the comments preceding each rung for additional information.

Adding to File 2

Ladder Rungs

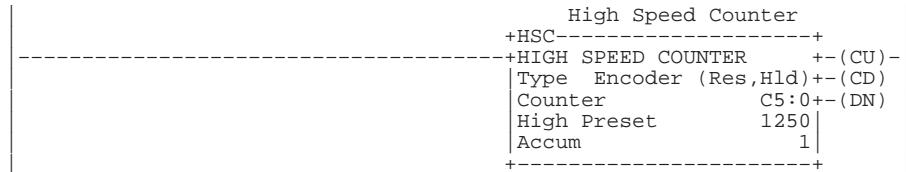
Rung 2:0

Initializes the high-speed counter each time that the RRUN mode is entered. The high-speed counter data area (N7:5 - N7:9) was chosen to correspond with the starting address (source address) of our HSL instruction. Note that the HSC instruction is disabled each entry into the RRUN mode until the first time that it is executed as true. (The high preset was "pegged" on initialization to prevent a high preset interrupt from occurring during the initialization process.)

| | | |
|---------------|--|--|
| 1'st Pass | | Output Mask (only use bit 0 ie. O:0/0) |
| S:1 | +MOV-----+ | +MOV-----+ |
| -----] [----- | +MOVE | +MOVE |
| 15 | Source 1 | Source 1 |
| | Dest N7:5 | Dest N7:5 |
| | 0 | 0 |
| | +-----+ | +-----+ |
| | High Output Pattern (turn off O:0/0) | High Output Pattern (turn off O:0/0) |
| | +MOV-----+ | +MOV-----+ |
| | +MOVE | +MOVE |
| | Source 0 | Source 0 |
| | Dest N7:6 | Dest N7:6 |
| | 0 | 0 |
| | +-----+ | +-----+ |
| | High Preset Value (counts to next hole) | High Preset Value (counts to next hole) |
| | +MOV-----+ | +MOV-----+ |
| | +MOVE | +MOVE |
| | Source 32767 | Source 32767 |
| | Dest N7:7 | Dest N7:7 |
| | 0 | 0 |
| | +-----+ | +-----+ |
| | Low output pattern (turn on O:0/0 each reset) | Low output pattern (turn on O:0/0 each reset) |
| | +MOV-----+ | +MOV-----+ |
| | +MOVE | +MOVE |
| | Source 1 | Source 1 |
| | Dest N7:8 | Dest N7:8 |
| | 0 | 0 |
| | +-----+ | +-----+ |
| | Low preset value (cause low preset int at reset) | Low preset value (cause low preset int at reset) |
| | +MOV-----+ | +MOV-----+ |
| | +MOVE | +MOVE |
| | Source 0 | Source 0 |
| | Dest N7:9 | Dest N7:9 |
| | 0 | 0 |
| | +-----+ | +-----+ |
| | High Speed Counter | High Speed Counter |
| | +HSL-----+ | +HSL-----+ |
| | +--+HSC LOAD | +--+HSC LOAD |
| | Counter C5:0 | Counter C5:0 |
| | Source N7:5 | Source N7:5 |
| | Length 5 | Length 5 |
| | +-----+ | +-----+ |

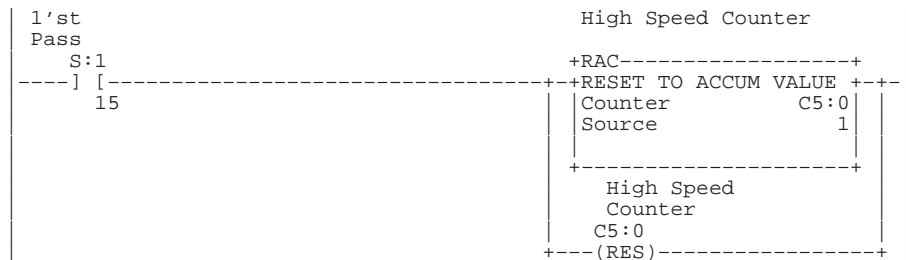
Rung 2:1

This HSC instruction is not placed in the high-speed counter interrupt subroutine. If this instruction were placed in the interrupt subroutine, the high-speed counter could never be started or initialized (because an interrupt must first occur in order to scan the high-speed counter interrupt subroutine).



Rung 2:2

This rung forces a high-speed counter low preset interrupt to occur each RRUN mode entry. An interrupt can only occur on the transition of the high-speed counter accum to a preset value (accum reset to 1, then 0). This is done to allow the high-speed counter interrupt subroutine sequencers to initialize. The order of high-speed counter initialization is: (1)load high-speed counter parameters (2)execute HSL instruction (3)execute true HSC instruction (4)(optional) force high-speed counter interrupt to occur.



Instruction List

File 2, Rung 0

Initializes the high-speed counter each time the RRUN mode is entered. The high-speed counter data area (N7:5 - N7:9) corresponds with the starting address (source address) of the HSL instruction. The HSC instruction is disabled each entry into the RRUN mode until the first time that it is executed as true. (The high preset was "pegged" on initialization to prevent a high preset interrupt from occurring during the initialization process.)

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|----------------|--------|
| 20 | -] [- | LD | 1'st Pass | | |
| | | | S1/15 | 0 | |
| 106 | | MOV | SRC | 1 0000H | |
| | | | Output Mask (only use bit 0 ie. 0:0/0) | | |
| | | | DEST N5 | | |
| 106 | | MOV | SRC | 0 0000H | |
| | | | High Output Pattern (turn off 0:0/0) | | |
| | | | DEST N6 | | |
| 106 | | MOV | SRC | 32767 0000H | |
| | | | High Preset Value (counts to next hole) | | |
| | | | DEST N7 | | |
| 106 | | MOV | SRC | 1 0000H | |
| | | | Low output pattern (turn on 0:0/0 each reset) | | |
| | | | DEST N8 | | |
| 106 | | MOV | SRC | 0 0000H | |
| | | | Low preset value (cause low preset intat reset) | | |
| | | | DEST N9 | | |
| 171 | | HSL | High Speed Counter | | |
| | | | CNTR C0 | | |
| | | | Output Mask (only use bit 0 ie. 0:0/0) | | |
| | | | SRC N5 | | |
| | | | LEN | 5 | |

File 2, Rung 1

This HSC instruction is not placed in the high-speed counter interrupt subroutine. If this instruction were placed in the interrupt subroutine, the high-speed counter could never be started or initialized (because an interrupt must first occur in order to scan the high-speed counter interrupt subroutine).

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------------------|--------|
| 170 | | HSC | TYPE | Encoder (Res,Hld) | |
| | | | High Speed Counter | 1250 | |
| | | | CNTR C0 | 0000H | |
| | | | PRE | | |
| | | | ACC | | |

File 2, Rung 2

Forces a high-speed counter low preset interrupt to occur each RRUN mode entry. An interrupt can only occur on the transition of the high-speed counter accum to a preset value (accum reset to 1, then 0). This is done to allow the high-speed counter interrupt subroutine sequencers to initialize. The order of high-speed counter initialization is: (1)load high-speed counter parameters (2)execute HSL instruction (3)execute true HSC instruction (4)(optional) force high-speed counter interrupt to occur.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|-------------------------------|-------|--------|
| 20 | -] [- | LD | 1'st Pass | | |
| | | | S1/15 | 0 | |
| 132 | | RAC | High Speed Counter CNTR C0 | | |
| | | | SRC | 1 | |
| 7 | | RES | High Speed Counter C0 | | |

Rungs 2.0 and 2.2 are required to write several parameters to the high-speed counter data file area. These two rungs are conditioned by the first pass bit during one scan when the controller is going from RPRG to RRUN mode.

The high-speed counter is used to control the conveyer position. It counts pulses supplied by the conveyer's encoder via hardware inputs I/0 and I/1. Hardware inputs I/2 (reset) and I/3 (hold) are connected to a photo-switch ensuring the HSC instruction only counts encoder pulses when a manual is in front of the drill and that the high-speed counter is reset at the leading edge of each manual.

The high-speed counter clears the conveyer drive output bit (O/0) each time a high preset is reached. As a result, the drive decelerates and stops the conveyer motor. The high-speed counter clears the output within microseconds ensuring accuracy and repeatability.

The high-speed counter sets the conveyer drive output bit (O/0) each time a low preset is reached. As a result, the drive accelerates and maintains the conveyer motor.

When the manual has travelled the specified distance set by the high-speed counter high preset value, the high-speed counter interrupt subroutine signals the main program to perform the drilling sequence. For more information regarding the interrupt subroutine used in this program, refer to the application example in chapter 13.

This example uses the Quadrature Encoder with reset and hold instruction. The high-speed counter accumulator increments and decrements based on the quadrature relationship of the encoder's A and B inputs (I/0 and I/1). The accumulator is cleared to zero when the reset is activated or when the RES instruction is executed.

All presets are entered as a relative offset to the leading edge of a manual. The presets for the hole patterns are stored in the SQO instructions. (Refer to chapter 13 for the SQO instruction.)

The high-speed counter external reset input (I/2) and the external hold input (I/3) are wired in parallel to prevent the high-speed counter from counting while the reset is active.

The input filter delays for both the high-speed counter A and B inputs (I/0 and I/1) as well as the high-speed counter reset and hold inputs (I/2 and I/3) can be adjusted. Refer to chapter 18 for more information on adjusting filters.

Adding to File 4

Ladder Rungs

```
Rung 4:5
| interrupt occurred due to low preset reached |
| C5:0 |-----+RET-----+ |
|----] [-----+RETURN |
| IL |-----+-----+ |
```

Rung 4:6
This rung signals the main program (file 2) to initiate a drilling sequence. The high-speed counter has already stopped the conveyor at the correct position using its high preset output pattern data (clear 0:0/0). This occurred within microseconds of the high preset being reached (just prior to entering this high-speed counter interrupt subroutine). The main program resets the drill sequence start bit and sets the conveyor drive bit (0:0/0) upon completion of the drilling sequence.

```
| interrupt occurred | Drill Sequence Start |
| due to high preset reached | |
| C5:0 |----- B3 |
|----] [----- (L)----- |
| IH |----- 32 |
```

```
Rung 4:7
|-----+END+-----|
```

Instruction List

File 4, Rung 5

Interrupt occurred due to low preset reached.

| FUN | GRAPHIC | MNEMONIC | PARAMETER | | VALUE | FORCES |
|------|---------|----------|-----------|--|-------|--------|
| CODE | SYMBOL | | NAME | ADDRESS | | |
| ---- | ----- | ----- | ---- | ----- | ----- | ----- |
| 20 | -] [- | LD | | | | |
| | | | | interrupt occurred due to low preset reached | | |
| | | | | C0/IL | 0 | |
| 134 | | RET | | | | |

File 4, Rung 6

Signals the main program (file 2) to initiate a drilling sequence. The high-speed counter has already stopped the conveyor at the correct position using its high preset output pattern data (clear O:0/0). This occurred within microseconds of the high preset being reached (just prior to entering this high-speed counter interrupt subroutine). The drill sequence subroutine resets the drill sequence start bit and sets the conveyor drive bit (O:0/0) upon completion of the drilling sequence.

| FUN | GRAPHIC | MNEMONIC | PARAMETER | | VALUE | FORCES |
|------|---------|----------|-----------|---|-------|--------|
| CODE | SYMBOL | | NAME | ADDRESS | | |
| ---- | ----- | ----- | ---- | ----- | ----- | ----- |
| 20 | -] [- | LD | | | | |
| | | | | interrupt occurred due to hi preset reached | | |
| | | | | C0/IH | 0 | |
| 41 | -(L)- | SET | | | | |
| | | | | Drill Sequence Start | | |
| | | | | B/32 | 0 | |

Using Communication Protocols

This chapter contains information about communication and the message (MSG) instruction. Specifically, this chapter contains information on:

- types of communication
- what the MSG instruction symbol looks like
- typical execution time for the MSG instruction
- how to use the MSG instruction
- application examples and timing diagrams

Important: Only Series B or later MicroLogix 1000 HHPs and Series C or later MicroLogix 1000 controllers support the MSG instruction. Only Series C MicroLogix 1000 HHPs fully support Series D functionality.

Only Series C MicroLogix 1000 HHPs will support MicroLogix 1000 analog controllers.

You cannot use the MicroLogix 1000 HHP to select or configure DF1 half-duplex slave communications in Series D discrete or Series A analog MicroLogix 1000 controllers.

Message Instruction

| Mnemonic | Function Code | Name | Purpose | Page |
|----------|---------------|--------------------|--|------|
| MSG | 200 | Message Read/Write | This instruction transfers data from one node to another via the communication port. When the instruction is enabled, the message is sent to a communication buffer. Replies are processed at the end of scan. | 15-2 |

Types of Communication

Communication is the ability of a device to send data or status to other devices. This capability typically falls into one of two categories: initiator (master) or responder (slave). Each of these are described below:

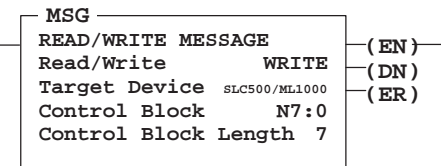
Initiator (Master) Communication

Initiator products can begin communication processes, which includes requesting information from other devices (reading) or sending information to other products (writing). In addition, initiator products are usually capable of replying to other devices when they make requests to read information. The Series C or later MicroLogix 1000 discrete controllers and the Series A or later MicroLogix 1000 analog controllers are in this class.

Initiator products can begin communication processes with other initiator products (peer-to-peer communication) or with responder (slave) products (initiator-to-responder communication).

Message Instruction (MSG)

Ladder representation:



Execution Times (µsec) when:

| True | False |
|------------------|-------|
| 180 ^① | 48 |

^① This only includes the amount of time needed to set up the operation requested. It does not include the time it takes to service the actual communication, as this time varies with each network configuration. As an example, 144ms is the actual communication service time for the following configuration: 3 nodes on DH-485 (2=MicroLogix 1000 programmable controllers and 1=PLC-500 A.I. Series™ programming software), running at 19.2K baud, with 2 words per transfer.

Responder (Slave) Communication

Responder products can only reply to other products. These devices are not capable of initiating an exchange of data; they only reply to requests made from initiator products. The Series A and B MicroLogix 1000 controllers are in this class.

The MSG is an output instruction that allows the controller to initiate an exchange of data with other devices. The relationship with the other devices can be either peer-to-peer communication or master-to-slave communication. The type of communication required by a particular application determines the programming configuration requirements of the MSG instruction.

Entering Parameters

When entering a message instruction, the following parameters need to be entered:

- **Read/Write** – read indicates that the local processor (processor in which the instruction is located) is receiving data; write indicates that the processor is sending data.
- **Target Device** – identifies the type of command used to establish communication. The target device can be a MicroLogix 1000 controller or SLC family processor using SLC commands, or a common interface file by selecting the CIF (Common Interface File) format. Valid options are:
 - SLC500/ML1000 – Allows communication between a MicroLogix 1000 controller and any other MicroLogix 1000 controller or SLC 500 family processor.
 - CIF – (Common Interface File) Allows communication between a MicroLogix 1000 controller and a non-MicroLogix 1000/SLC 500 device. The CIF data is automatically delivered to integer file 9 in SLC 500 processors or integer file 7 in MicroLogix 1000 controllers. The CIF protocol is also used for PLC-2 type messages.
- **Control Block Address** – an integer file address that you select. It consists of seven integer words, containing the status bits, target file address, and other data associated with the MSG instruction.
- **Control Block Length** – fixed at seven elements. This field cannot be altered.
- **Target Node** – the node address of the target device. Valid entries are 0–254 for DF1; 0–31 for DH-485.
- **Local Address** – the address of the local device.
- **Target Address** – If SLC500/ML1000 is selected as the target device, a target address must be entered.
- **Target Offset** – If CIF is selected as the target device, you must enter a value for the offset into the CIF.
- **Message Length** – specifies the length of the message instruction in elements. 1-word elements are limited to 1–41; 3-word elements are limited to 1–13.

Important: When running a MicroLogix 1000 program on an SLC 5/03 or SLC 5/04 processor, the MSG control block length increases from 7 to 14 words. If you plan to run a MicroLogix 1000 program with one of these processors, make sure that the program has at least 7 unused words following each MSG control block.

The following table illustrates combinations of message types and target devices and their valid file types.

| Command Type | Message Type | Initiating Device | Valid File Types | Target Device ^{①②③} | Valid File Types |
|---------------|--------------|-------------------|------------------|------------------------------|---|
| SLC500/ML1000 | Write | MicroLogix 1000 | O,I,S,B,T,C,R,N | MicroLogix 1000 | O,I,S,B,T,C,R,N |
| SLC500/ML1000 | Read | MicroLogix 1000 | O,I,S,B,T,C,R,N | MicroLogix 1000 | O,I,S,B,T,C,R,N |
| CIF | Write | MicroLogix 1000 | O,I,S,B,T,C,R,N | MicroLogix 1000 | N7 |
| CIF | Read | MicroLogix 1000 | O,I,S,B,T,C,R,N | MicroLogix 1000 | N7 |
| SLC500/ML1000 | Write | MicroLogix 1000 | O,I,S,B,T,C,R,N | SLC 500 | O ^④ ,I ^④ ,S,B,T,C,R,N |
| SLC500/ML1000 | Read | MicroLogix 1000 | O,I,S,B,T,C,R,N | SLC 500 | O ^④ ,I ^④ ,S,B,T,C,R,N |
| CIF | Write | MicroLogix 1000 | O,I,S,B,T,C,R,N | SLC 500 | N9 |
| CIF | Read | MicroLogix 1000 | O,I,S,B,T,C,R,N | SLC 500 | N9 |

- ① The DF1 Full-Duplex protocol can be used if the target device supports it. Such devices include MicroLogix 1000 controllers (any series), SLC 5/03, SLC 5/04 and SLC 5/05 processors, and PLC-5 processors (CIF command type only).
- ② The DH-485 protocol can be used if the target device supports it. Such devices include MicroLogix 1000 controllers (except for Series A and B discrete controllers) and SLC 500, SLC 5/01, SLC 5/02, SLC 5/03, SLC 5/04 or SLC 5/05 processors.
- ③ The DF1 Half-Duplex protocol can also be used with Series D or later discrete and all analog MicroLogix 1000 controllers, but a master is required, such as an SLC 5/03, SLC 5/04 or SLC 5/05 processor.
- ④ SLC 500, SLC 5/01, and SLC 5/02 processors do not support O or I file access from a MSG instruction. SLC 5/03, SLC 5/04 and SLC 5/05 processors do support O and I file access, but only when unprotected.

Control Block Layout

The control block layouts shown below illustrate SLC500/ML1000 type messages.

Control Block Layout – SLC500/ML1000

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | Word | | | | | |
|------------------------------------|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|-------------|---|--|--|--|---|
| EN ST DN ER | | | | | | | | | | | EW NR TO | | | | | Error Code | | | | | 0 |
| | | | | | | | | | | | | | | | | Node Number | | | | | 1 |
| Reserved for length (in elements) | | | | | | | | | | | | | | | | | 2 | | | | |
| File Number | | | | | | | | | | | | | | | | | 3 | | | | |
| File Type (O, I, S, B, T, C, R, N) | | | | | | | | | | | | | | | | | 4 | | | | |
| Element Number | | | | | | | | | | | | | | | | | 5 | | | | |
| Subelement Number | | | | | | | | | | | | | | | | | 6 | | | | |

Control Block Layout – CIF

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | Word |
|-----------------------------------|----|----|----|----|----|----|----|----------|----|------------|----|-------------|----|----|----|------|
| EN ST DN ER | | | | | | | | EW NR TO | | Error Code | | | | | | 0 |
| | | | | | | | | | | | | Node Number | | | | 1 |
| Reserved for Length (in elements) | | | | | | | | | | | | | | | | 2 |
| Offset Bytes | | | | | | | | | | | | | | | | 3 |
| Not used | | | | | | | | | | | | | | | | 4 |
| Not used | | | | | | | | | | | | | | | | 5 |
| Not used | | | | | | | | | | | | | | | | 6 |

Using Status Bits

```

Read/Write:                READ                ignore if timed out:  0  TO
Target Device:             SLC500/ML1000        to be retried:       0  NR
Control Block:             N7:0                awaiting execution:   0  EW
Local Destination File Address:  ***
Target Node:               0                    error:               0  ER
Target File Address:       ***                    message done:        0  DN
Message Length in elements  ***                    message transmitting: 0  ST
                                                                message enabled:     0  EN

                                                                control bit address:  N7:0/8

ERROR CODE:  0
Error Code Desc:

```

MSG Instruction Status Bits

The right column in the display above lists the various MSG instruction status bits. These are explained below:

- **Time Out Bit TO (bit 08)** Temporarily set this bit (1) to clear an existing MSG instruction. This bit has no effect unless the ST bit has first been set due to receiving an ACK (acknowledge). Your application must supply its own timeout value. This bit is reset on any false-to-true rung transition.
 - **Negative Response Bit NR (bit 09)** is set if the target processor is responding to your message, but can not process the message at the present time. The NR bit is reset at the next false-to-true MSG rung transition that has a transmit buffer available. It is used to determine when to send retries. The ER bit is also set at this time. Use this feedback to initiate a retry of your message at a later time. This bit is used with DH-485 protocol only.
 - **Enabled and Waiting Bit EW (bit 10)** is set on any false-to-true MSG rung transition. This bit is reset when an ACK or NAK (no acknowledge) is received, or on any true-to-false MSG rung transition.
- Important:** The operation of the EW bit has changed since Series C.
- **Error Bit ER (bit 12)** is set when message transmission has failed. The ER bit is reset the next time the rung goes from false to true.

- **Done Bit DN (bit 13)** is set when the message is transmitted successfully. The DN bit is reset (cleared) the next time the rung goes from false to true.
- **Start Bit ST (bit 14)** is set when the processor receives acknowledgement from the target device. This identifies that the target device has started to process the MSG request. The ST bit is reset when the DN, ER, or TO bit is set or on a false-to-true rung transition.
- **Enable Bit EN (bit 15)** is set only if the transmit buffer is available. If the transmit buffer is not available, the EN flag remains false. When the transmit buffer becomes available, the EN flag goes true. It remains set until the next false rung execution after the MSG completes (DN bit set) or an error occurs (ER bit set).

Important: The operation of the EN bit has changed since Series C.

The operation associated with a message read or write instruction is performed when you enable the instruction. Replies are processed at the end of the scan.

Controller Communication Status Bit

When using the MSG instruction, you should also use the following controller communication status bit:

Active Protocol Bit (S:0/11) – This is a read-only bit that indicates which communication protocol is currently enabled or functioning: where 0 = DF1 (default) and 1 = DH-485. Use this bit in your program to restrict message operation to a specific protocol.

Entering the Instruction

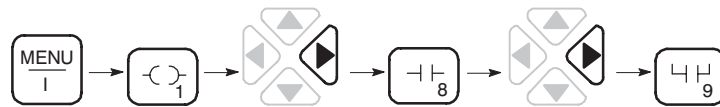
You enter the instruction from within the program monitor functional area. The following items apply when entering the instruction:

Important: Only Series B or later MicroLogix 1000 HHPs and Series C or later MicroLogix 1000 controllers support the MSG instruction. Only Series C MicroLogix 1000 HHPs fully support Series D functionality.

Only Series C MicroLogix 1000 HHPs will support MicroLogix 1000 analog controllers.

In order to enter the instruction, your program must be configured to operate with Series C or later MicroLogix 1000 controllers. See page 18–18 for more information.

- Whenever you see asterisks on the display, the HHP is waiting for data entry (i.e., a number).
- When entering the target address, an element or subelement is entered by first pressing the right arrow key. For example, to enter the address I1:8.9, you would press the following key sequence:



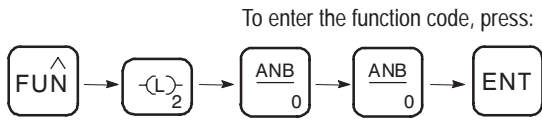
- If you see a down arrow on the display it means there are more options available. To scroll through the options, press this key:



- You can return to previously entered operands by pressing this key:



Then if you want to edit that operand, press **DEL** or **FUN-DEL** and enter new parameters. Press **ENT** to accept the operand and move on to the next one. Once the entire instruction is entered, if you want to edit the instruction's parameters, you must go into the overwrite mode. (See page 17–4.)



```
P 0 0 8  † M S G      T Y P E
W R I T E ↓
```

Select Read or Write

```
P 0 0 8  † M S G      D E V
S L C 5 0 0 / M L ↓
```

Select SLC500/ML1000 or CIF

```
P 0 0 8  † M S G      C B L K
N 0
```

Enter Control Block Address

```
P 0 0 8  † M S G      C B L N
7
```

Enter Control Block Length (Set at 7)

```
P 0 0 8  † M S G      N O D E
* * * * * 5
```

Identify Target Node

```
P 0 0 8  † M S G      L O C A L
N 1 5
```

Enter Local Address

```
P 0 0 8  † M S G      T A R G
N 2 5
```

Enter Target Address

```
P 0 0 8  † M S G      L E N
* * * * * 2
```

Enter Data Length in Elements

Note: →

When entering the target file type, the MicroLogix default file number is automatically inserted. If you want to change the file number, simply press the desired number.

Once the instruction is entered, the parameters are displayed as six screens, as shown here:

| | | | | |
|-------------------|---|-----------|--|---------|
| P 0 0 8 | ┆ | M S G | | T Y P E |
| S L C 5 0 0 / M L | | W R I T E | | |

| | | | | |
|---------|---|-------|--|-----------|
| P 0 0 8 | ┆ | M S G | | C B L K |
| N 0 | | | | 0 0 0 0 H |

| | | | | |
|---------|---|-------|--|---------|
| P 0 0 8 | ┆ | M S G | | N O D E |
| | | | | 5 |

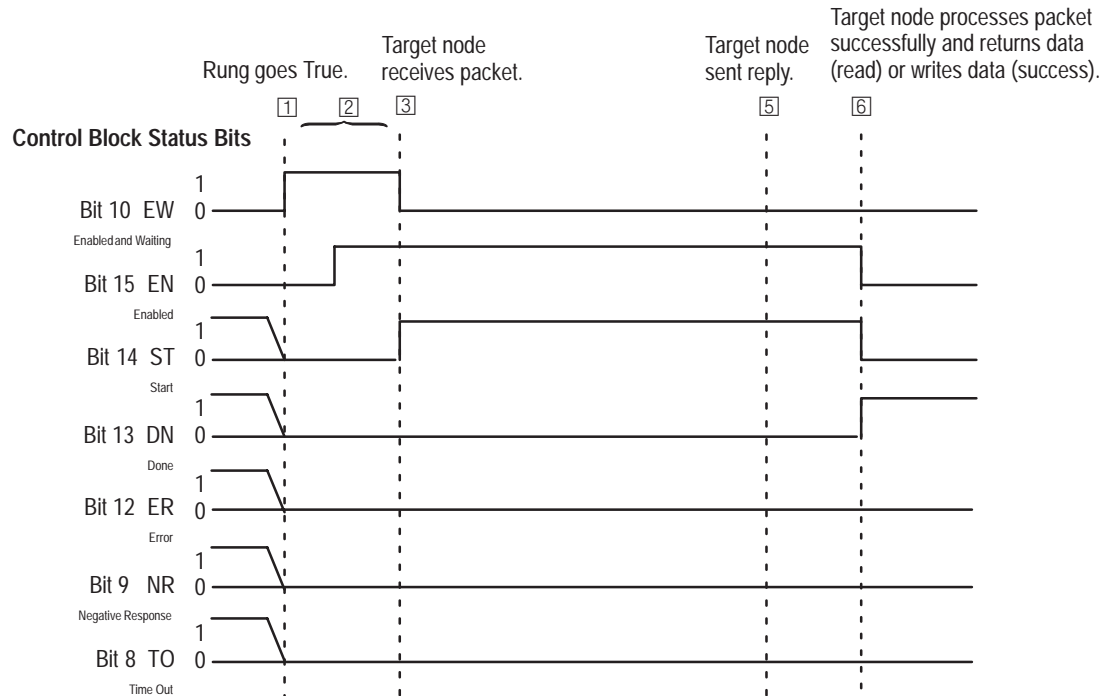
| | | | | |
|---------|---|-------|--|-----------|
| P 0 0 8 | ┆ | M S G | | L O C A L |
| B 7 | | | | 0 0 0 0 H |

| | | | | |
|---------|---|-------|--|-------------|
| P 0 0 8 | ┆ | M S G | | T A R G |
| | | | | R 9 1 : 8 2 |

| | | | | |
|---------|---|-------|--|-------|
| P 0 0 8 | ┆ | M S G | | L E N |
| | | | | 2 |

Timing Diagram for a Successful MSG Instruction

The following section illustrates a successful timing diagram for a MSG instruction in a Series D or later MicroLogix 1000 discrete controller or a MicroLogix 1000 analog controller.



- ① The EW bit is set (1) and the ST, DN, NR, and TO flags are cleared. If the transmit buffer is not available, the EN flag remains false (0).
- ② When rung conditions go true and the transmit buffer becomes available, the EN flag goes true (1). The EN bit remains set until either the DN, ER, or TO bit is set. The TO bit has no effect unless the ST bit has first been set.
- ③ If the Target Node successfully receives the MSG packet, it sends back an ACK (acknowledge). The ACK causes the processor to clear bit S:2/7. (Bit S:2/7 is valid for Series C discrete only). Note that the Target Node has not yet examined the MSG packet to see if it understands your request. It is replying to the initial connection.

At the next end of scan, the EW bit is cleared (0) and the ST bit is set (1). Once the ST bit is set, the processor waits indefinitely for a reply from the Target Node. The Target Node is not required to respond within any given time frame. At this time, no other MSG instruction is serviced.

Note: If the Target Node faults or power cycles during the time frame after the ST bit is set and before the reply is returned, you will never receive a reply. No other MSG instructions are able to be serviced unless this MSG is terminated in error using the TO bit. This is why it is recommended you use a timer in conjunction with the TO bit to clear any pending instructions. (When the TO bit is set [1] it clears pending messages.) Typically message transactions are completed within a couple of seconds. It is up to the programmer to determine how long to wait before clearing the buffer and then re-transmitting.

Step 4 is not shown in the timing diagram.

- ④ If you do not receive an ACK, step 3 does not occur. Instead a NAK (no acknowledge) is received. When this happens, the ST bit remains clear. A NAK indicates:

- the Target Node is too busy, or
- it received a MSG packet with a bad checksum.

No response indicates:

- either the Target Node is not there, or
- it does not respond because the MSG packet was corrupted in transmission.

When a NAK occurs, the EW bit is cleared. (Note that the NR bit will only be set for DH-485 and NAK conditions. An error code 02H, Target Node is busy, is received which causes the NR bit to be set.) The ER bit is also set which indicates that the MSG instruction failed.

Monitor the NR bit. If it is set, indicating that the Target Node is busy, you may want to initiate some other process (e.g., an alarm or a retry later). The NR bit is cleared when the rung logic preceding the MSG changes from false to true.

- ⑤ When an ACK occurs, the Target Node sends one of three responses shown in Step 6.
- ⑥ Following the successful receipt of the packet, the Target Node sends a reply packet. The reply packet will contain one of the following responses:
- I have successfully performed your write request.
 - I have successfully performed your read request, and here is your data.
 - I have not performed your request because of an error.

At the next end of scan, following the Target Node's reply, the MicroLogix 1000 controller examines the MSG packet from the target device. If the reply contains "I have successfully performed your write request," the DN bit is set and the ST bit is cleared. The MSG instruction is complete. If the MSG rung is false, the EN bit is cleared the next time the MSG instruction is scanned.

If the reply contains "I have successfully performed your read request, and here is your data," the data is written to the appropriate data table, the DN bit is set, and the ST bit is cleared. The MSG instruction function is complete. If the MSG rung is false, the EN bit is cleared the next time the MSG instruction is scanned.

If the reply contains "I have not performed your request, because of an error," the ER bit is set and the ST bit is cleared. The MSG instruction function is complete. If the MSG rung is false, the EN bit is cleared the next time the MSG instruction is scanned.

MSG Instruction Error Codes

Any MSG instruction that is in progress during a network protocol switch is not processed and is discarded. For more information on network protocol switching, see page 3–12.

When an error condition occurs, the error code is stored in the *lower byte* of the first control word assigned to the MSG instruction.

| Error Code | Description of Error Condition |
|------------------|---|
| 02H | Target node is busy. |
| 03H | Target node cannot respond because message is too large. |
| 04H | Target node cannot respond because it does not understand the command parameters OR the control block may have been inadvertently modified. |
| 05H | Local processor is offline (possible duplicate node situation). |
| 06H | Target node cannot respond because requested function is not available. |
| 07H | Target node does not respond. |
| 08H | Target node cannot respond. |
| 09H | Local modem connection has been lost. |
| 0AH | Buffer unavailable to receive SRD reply. |
| 0BH | Target node does not accept this type of MSG instruction. |
| 0CH | Received a master link reset. |
| 10H | Target node cannot respond because of incorrect command parameters or unsupported command. |
| 15H | Local channel configuration parameter error exists. |
| 18H | Broadcast (Node Address 255) is not supported. |
| 1AH ^① | Target node cannot respond because another node is file owner (has sole file access). |
| 1BH ^① | Target node cannot respond because another node is program owner (has sole access to all files). |
| 37H | Message timed out in local processor. |
| 39H | Message was discarded due to a communication protocol switch. |
| 3AH | Reply from target is invalid. |
| 50H | Target node is out of memory. |
| 60H | Target node cannot respond because file is protected. |
| E7H | Target node cannot respond because length requested is too large. |
| EBH | Target node cannot respond because target node denies access. |
| ECH | Target node cannot respond because requested function is currently unavailable. |
| FAH | Target node cannot respond because another node is file owner (has sole file access). |
| FBH | Target node cannot respond because another node is program owner (has sole access to all files). |

^① Error codes 1A and 1B valid for Series C discrete only.

Note: For 1770–6.5.16 DF1 Protocol and Command Set users:

The MSG error code reflects the STS field of the reply to your MSG instruction. Codes E0 – EF represent EXT STS codes 0 – F. Codes F0 – FC represent EXT STS codes 10 – 1C.

Application Examples that Use the MSG Instruction

Example 1 – Continuously Writing Data from a MicroLogix Controller

In this example, a communication link is created between two Series C or higher MicroLogix 1000 discrete controllers, where one controller is writing data to another. The communication link is set up for continuous operation with automatic recovery. Performance in logic is as fast as possible, with the primary restriction being the communication link protocol (DH-485 or DF1 full-duplex) and the baud rate.

Although the DH-485 protocol is demonstrated here, this example can be used with either DH-485 or DF1 protocols. To run this on DF1, you would only need to change the active protocol bit (rung 2.2), the default primary protocol bit (S2:0/10 in the status file), and the message instruction variables. For maximum possible communication performance (throughput), select DF1 full duplex running at 38.4k baud.

Rung 2:0

This rung monitors the write message instruction for errors or lockup conditions and restarts communication whenever the link becomes valid. Lockup conditions are situations that may arise if the communication path is somehow corrupted. Some examples would be power lost at destination device or a cut cable. Error conditions are typically caused from noise on the communication link, faulty device(s) on the network, etc.

| Write Message No Response | Message Error Retry Timer | Message Retry Timer |
|------------------------------|------------------------------|----------------------------|
| N7:50 | T4:9 | +TON-----+ |
| 9 | DN | +TIMER ON DELAY +- (EN) +- |
| Write Message | | Timer T4:9+- (DN) |
| Error | | Time Base 0.01 |
| N7:50 | | Preset 100< |
| | | Accum 0< |
| 12 | | |
| Write Message | | |
| Start | | |
| N7:50 | | |
| 14 | | |

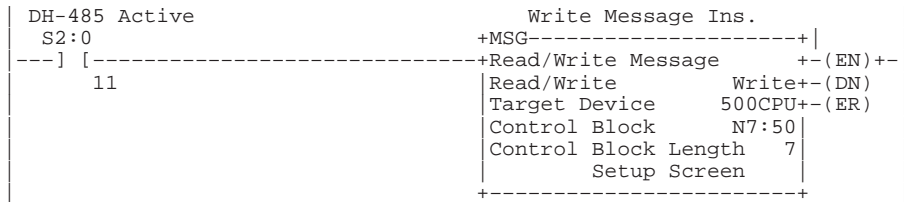
Rung 2:1

The Time Out bit (TO) associated with each message instruction is used to clear the controllers communication buffer and message instruction. Setting these bits basically places the controllers communication section in the same condition as when the controller power up. These bits allow control program to reset or recover from unexpected events (e.g., errors, power problems, media problems).

| Message Error Retry Timer | Write Message Time Out |
|------------------------------|---------------------------|
| T4:9 | N7:50 |
| DN | () |
| | 8 |

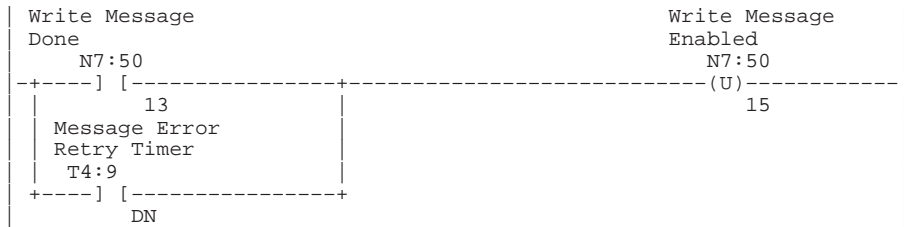
Rung 2:2

Write message with preceding logic. It is STRONGLY recommended that bit S2:0/11 (active protocol bit) be used to condition all message instructions. This bit only allows the message instruction to operate when the correct protocol is active. NOTE: If DF1 was the protocol used by the message instruction, use -]/[- S2:0/11 for the preceding logic.

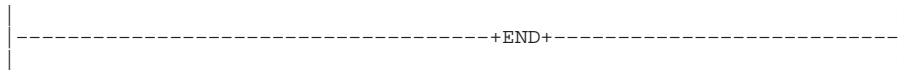


Rung 2:3

This rung monitors the message instruction for "Done" conditions. When the message is done, the enable bit is cleared allowing communication to run again. The error retry bit is also used to reset the message if an error or lockup condition is encountered.



Rung 2.4



Example 2 – Continuously Reading Data from a MicroLogix Controller

In this example, a communication link is created between two Series C or higher MicroLogix 1000 discrete controllers, where one controller is reading data from another. The communication link is set-up for continuous operation with automatic recovery. Performance in logic is as fast as possible, with the primary restriction being the communication link protocol (DH-485 or DF1 full-duplex) and the baud rate.

Although the DH-485 protocol is demonstrated here, this example can be used with either DH-485 or DF1 protocols. To run this on DF1, you would only need to change the active protocol bit (rung 2.2), the default primary protocol bit (S2:0/10 in the status file), and the message instruction variables. For maximum possible communication performance (throughput), select DF1 full duplex running at 38.4k baud.

Rung 2:0

This rung monitors the read message instruction for errors or lockup conditions and restarts communication whenever the link becomes valid. Lockup conditions are situations that may arise if the communication path is somehow corrupted. Some examples would be power lost at destination device, or a cut cable. Error conditions are typically caused from noise on the communication link, faulty device(s) on the network, etc.

| Read Message No Response N7:60 | Message Error Retry Timer T4:9 | Message Retry Timer |
|--------------------------------------|--------------------------------------|--|
| 9 | DN | +TON-----+ +TIMER ON DELAY +- (EN) +- Timer T4:9+- (DN) Time Base 0.01 Preset 100< Accum 0< |
| 12 | | |
| 14 | | |

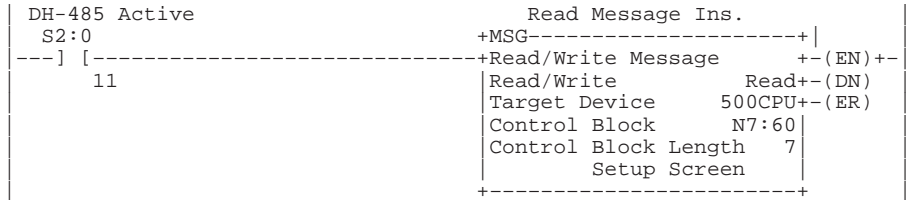
Rung 2:1

The Time Out bit (TO) associated with each message instruction is used to clear the controllers communication buffer and message instruction. Setting these bits basically places the controllers communication section in the same condition as when the controller power up. These bits allow control program to reset or recover from unexpected events (e.g., errors, power problems, media problems).

| Message Error Retry Timer T4:9 | Read Message Time Out N7:60 |
|--------------------------------------|-----------------------------------|
| DN | () |
| | 8 |

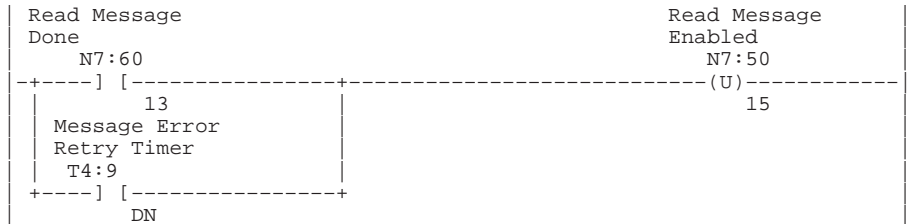
Rung 2:2

Read message with preceding logic. It is STRONGLY recommended that bit S2:0/11 (active protocol bit) be used to condition all message instructions. This bit only allows the message instruction to operate when the correct protocol is active. NOTE: If DF1 was the protocol used by the message instruction, use -]/[- S2:0/11 for the preceding logic.

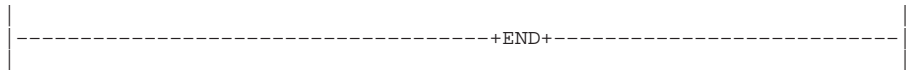


Rung 2:3

This rung monitors the message instruction for "Done" conditions. When the message is done, the enable bit is cleared allowing communication to run again. The error retry bit is also used to reset the message if an error or lockup condition is encountered.



Rung 2.4



Example 3 – Report on Exception/Change of State, Write Data from a MicroLogix Controller

In this example, a communication link is created between two Series C or higher MicroLogix 1000 discrete controllers, where one controller is writing data to another. The communication link is set-up for continuous operation with automatic recovery. Performance in logic is as fast as possible, with the primary restriction being the communication link protocol (DH-485 or DF1 full-duplex) and the baud rate.

Although the DH-485 protocol is demonstrated here, this example can be used with either DH-485 or DF1 protocols. To run this on DF1, you would only need to change the active protocol bit (rung 2.2), the default primary protocol bit (S2:0/10 in the status file), and the message instruction variables. For maximum possible communication performance (throughput), select DF1 full duplex running at 38.4k baud.

Rung 2:0

This rung monitors the write message instruction for errors or lockup conditions and restarts communication whenever the link becomes valid. Lockup conditions are situations that may arise if the communication path is somehow corrupted. Some examples would be power lost at destination device, or a cut cable. Error conditions are typically caused from noise on the communication link, faulty device(s) on the network, etc.

| Write Message | Message Error | Message Retry Timer |
|----------------------|---------------------|------------------------------|
| No Response N7:50 | Retry Timer T4:9 | +TON-----+ TIMER ON DELAY |
| 9 | DN | T4:9+-(DN) |
| Write Message | | Time Base 0.01 |
| Error | | Preset 1 |
| N7:50 | | Accum 0 |
| 12 | | |
| Write Message | | |
| Start | | |
| N7:50 | | |
| 14 | | |

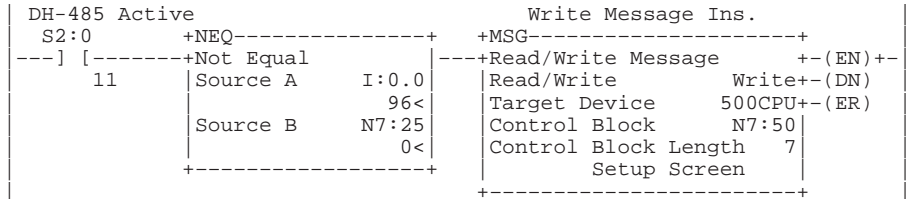
Rung 2:1

The Time Out bit (TO) associated with each message instruction is used to clear the controllers communication buffer and message instruction. Setting these bits basically places the controllers communication section in the same condition as when the controller power up. These bits allow control program to reset or recover from unexpected events (e.g., errors, power problems, media problems).

| Message Error | Write Message |
|---------------|---------------|
| Retry Timer | Time Out |
| T4:9 | N7:50 |
| DN | () |
| | 8 |

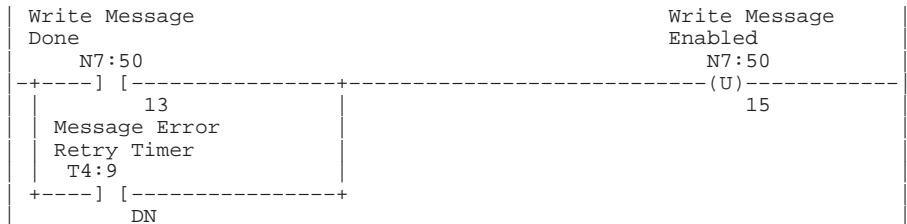
Rung 2:2

Write message with report by exception (RE) or change of state (COS) logic. COS is used to only transmit data/information when some type of change has occurred (time, data, etc.). This type of communications is extremely beneficial on networks because any time you can limit unnecessary traffic on a network, your ability to send information without delays is improved. It is STRONGLY recommended that bit S2:0/11 (active protocol bit) be used to condition all message instructions. This bit only allows the message instruction to operate when the correct protocol is active. NOTE: If DF1 was the protocol used by the message instruction, use -]/[- S2:0/11 for the preceding logic.



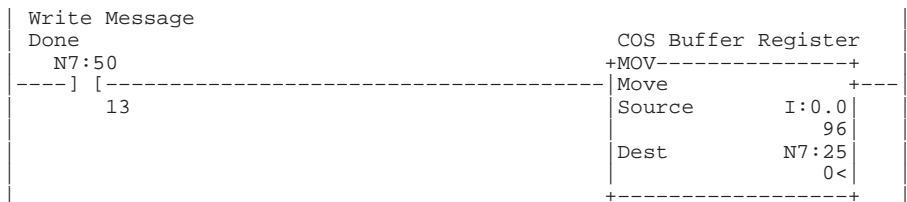
Rung 2:3

This rung monitors the message instruction for "Done" conditions. When the message is done, the enable bit is cleared allowing communication to run again. The error retry bit is also used to reset the message if an error or lockup condition is encountered.

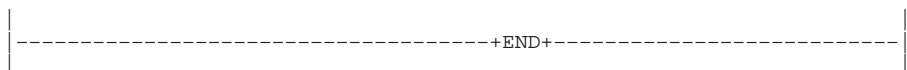


Rung 2:4

This MOV instruction is used to set N7:25 to I:0.0 after the message is successfully transmitted. This stops the message instruction in rung 2.2 from executing. If I:0.0 changes, the NEQ instruction in rung 2.2 triggers the message instruction to write data to the target device. (N7:25 is used in the comparison in rung 2.2 to enable or disable the message instruction from being processed.)



Rung 2.5



Example 4 – Continuously Reading and Writing Data with a MicroLogix Controller

In this example, a communication link is created between two Series C or higher MicroLogix 1000 discrete controllers. This example creates a master/slave communication link between two controllers (one controller reading and writing data to another controller) with a MicroLogix 1000 as the master. The communication link is set-up for continuous operation with automatic recovery. Performance in logic is as fast as possible, with the primary restriction being the communication link protocol (DH-485 or DF1 full-duplex) and the baud rate.

Although the DH-485 protocol is demonstrated here, this example can be used with either DH-485 or DF1 protocols. To run this on DF1, you would only need to change the active protocol bit (rungs 2.2 and 2.3), the default primary protocol bit (S2:0/10 in the status file), and the message instruction variables. For maximum possible communication performance (throughput), select DF1 full duplex running at 38.4k baud.

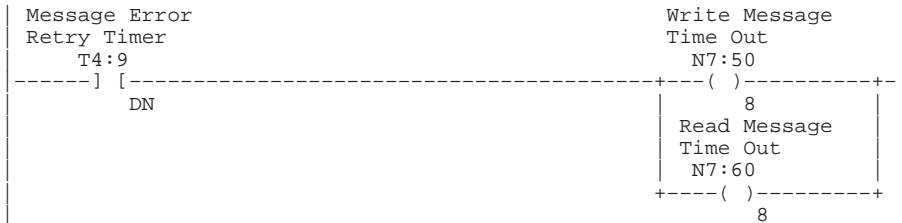
Rung 2:0

This rung monitors both message instructions for errors or lockup conditions and restarts communication whenever the link becomes valid. Lockup conditions are situations that may arise if the communication path is somehow corrupted. Some examples would be power lost at destination device or a cut cable. Error conditions are typically caused from noise on the communication link, faulty device(s) on the network, etc.

| Write Message No Response N7:50 | Message Error Retry Timer T4:9 | Message Retry Timer |
|---|--------------------------------------|--|
| +---] [-----] / [-----] +-----+ 9 Write Message Error N7:50 | | +TIMER ON DELAY +-(EN)+ Timer T4:9+-(DN) Time Base 0.01 Preset 100< Accum 0< |
| +---] [-----] 12 Write Message Start N7:50 | | |
| +---] [-----] 14 Read Message Error N7:60 | | |
| +---] [-----] 12 Read Message No Response N7:60 | | |
| +---] [-----] 9 Read Message Start N7:60 | | |
| +---] [-----] 14 | | |

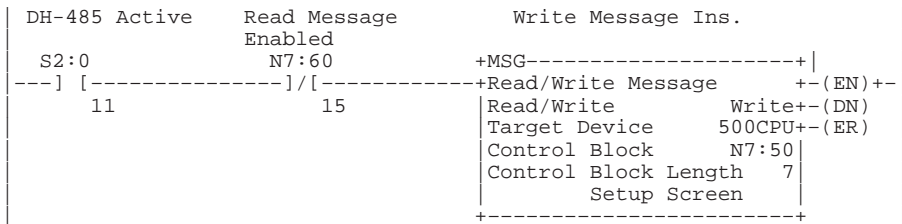
Rung 2:1

The Time Out bit (TO) associated with each message instruction is used to clear the controllers communication buffer and message instruction. Setting these bits, basically places the controllers communication section in the same condition as when the controller power up. These bits allow control program to reset or recover from unexpected events (e.g., errors, power problems, media problems).



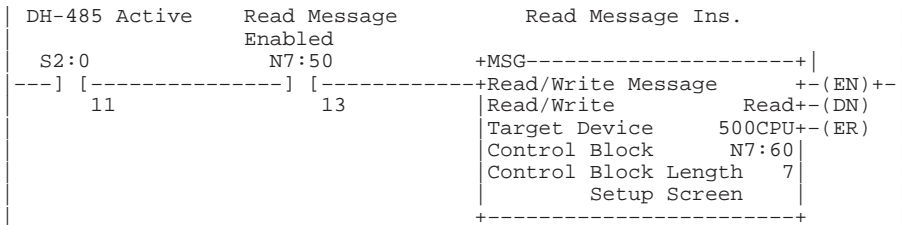
Rung 2:2

Write message with preceding logic. It is STRONGLY recommended that bit S2:0/11 (active protocol bit) be used to condition all message instructions. This bit only allows the message instruction to operate when the correct protocol is active. The read message enabled bit (N7:60/15) is used to "open" the rung of the write message when the read message is active. This sequences the write message to proper state, allowing immediate transmission when the read message completes its operation. NOTE: If DF1 was the protocol used by the message instruction, use -]/[- S2:0/11 for the preceding logic.



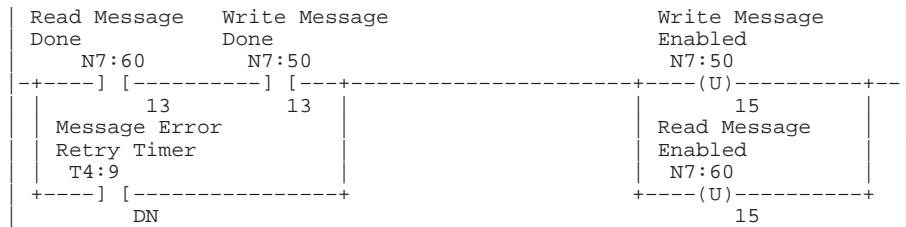
Rung 2:2

Write message with preceding logic. It is STRONGLY recommended that bit S2:0/11 (active protocol bit) be used to condition all message instructions. This bit only allows the message instruction to operate when the correct protocol is active. The write message enabled bit (N7:50/13) is used to enable the read message as soon as the write message completes its operation. NOTE: If DF1 was the protocol used by the message instruction, use -]/[- S2:0/11 for the preceding logic.

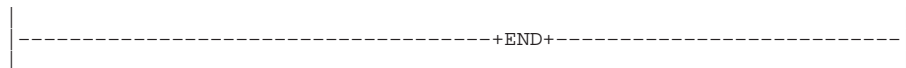


Rung 2:3

This rung monitors the message instruction for "Done" conditions. When both messages are done, both messages are reset, allowing the sequence to be restarted in the next scan. The error retry bit is also used to reset both messages if an error or lockup condition is encountered.



Rung 2.5



Instruction List Programming

This chapter uses programming examples to teach you instruction list programming. The chapter also lists programming considerations.

Programming Examples

In the section Applying Logic to Your Schematics, page 6–11, you learned the concepts behind ladder logic. You were also shown the instruction list (Boolean) equivalent of a simple rung of logic. This section builds on that by showing you some more rung examples and their equivalent instruction list(s).

Important: Although the rung examples shown here use only bit instructions, you can apply the methodology that is shown to any of the instructions you learned about in chapters 8–14.

Because of the flexibility of instruction list programming, there is often more than one instruction list representation possible for the same rung. However, there is always one representation that is considered to be the most optimized. This is the shortest list representation possible for the rung.

Important: For instruction list programming, you *must* begin every rung by pressing the key shown here:



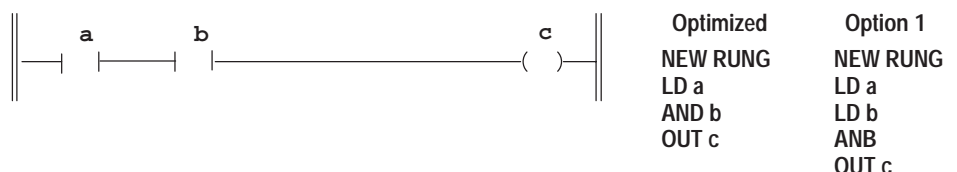
To help you remember to do this, each instruction list example in this chapter begins with the words NEW RUNG. (See page 17–2 for more information.)

Input Rung Examples

Examples of input rungs and their optimized instruction list representation are provided below. Since many of these rungs have more than one possible instruction list representation, optional representations are shown to the right of the optimized list.

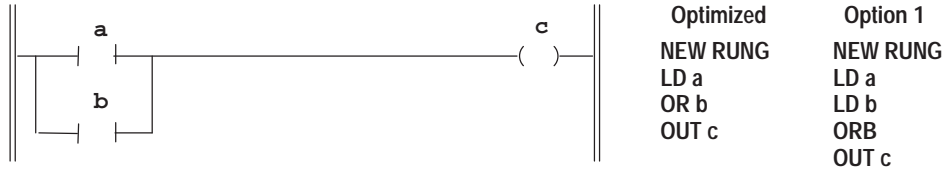
Series Inputs

This example shows the instruction list for a simple series connection.



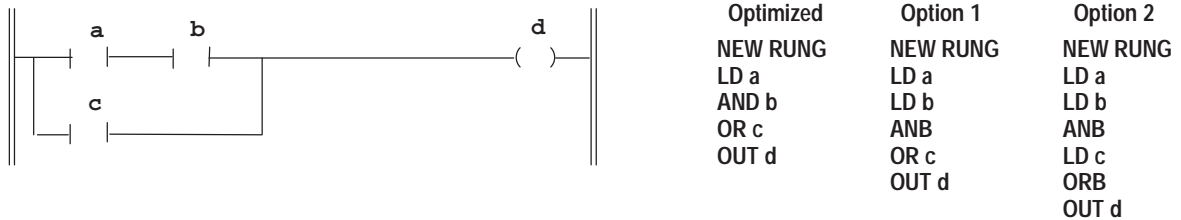
Parallel Inputs

When the inputs are in parallel, the instruction list representation is as follows:



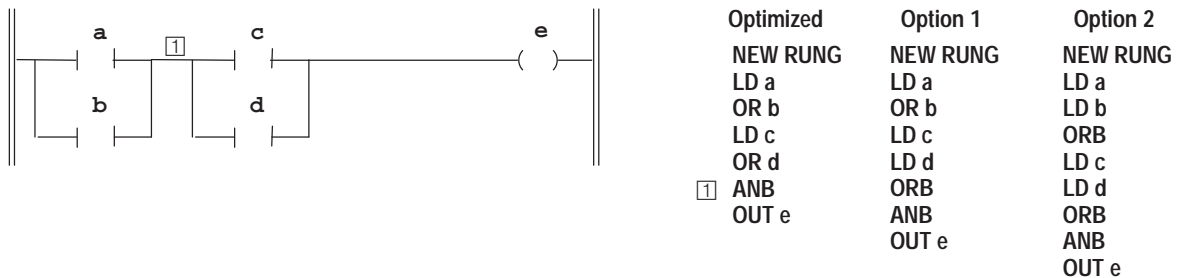
Parallel Input Branching

An input branch can be used to allow more than one combination of input conditions on parallel branches. An example of this type of rung is shown below.



Series Block Connection

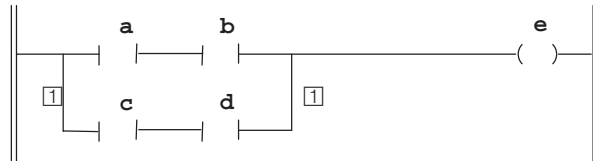
When two blocks of instructions are connected in series, an ANB instruction is used, as shown here:



(See page 8–12 for more information regarding the use of the ANB instruction.)

Parallel Block Connection

If two blocks of instructions are connected in parallel, an ORB instruction is used. An example of this type of block connection is provided below.

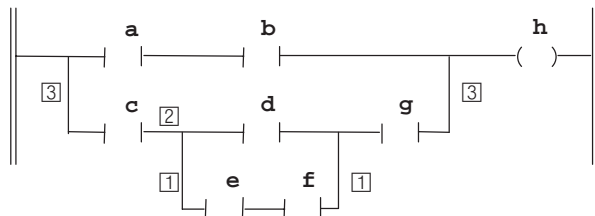


| Optimized | Option 1 | Option 2 |
|-----------|----------|----------|
| NEW RUNG | NEW RUNG | NEW RUNG |
| LD a | LD a | LD a |
| AND b | LD b | LD b |
| LD c | ANB | ANB |
| AND d | LD c | LD c |
| 1 ORB | AND d | LD d |
| OUT e | ORB | ORB |
| | OUT e | ORB |
| | | OUT e |

(See page 8–12 for more information regarding the use of the ORB instruction.)

Nested Input Branching

You can nest input branches to avoid using redundant instructions. This example shows how ANB and ORB instructions are used together in these type of rungs. (See page 8–12 for more information regarding the use of these instructions.)



| Optimized | Option 1 | Option 2 |
|-----------|----------|----------|
| NEW RUNG | NEW RUNG | NEW RUNG |
| LD a | LD a | LD a |
| AND b | LD b | LD b |
| LD c | ANB | ANB |
| LD d | LD c | LD c |
| LD e | LD d | LD d |
| AND f | LD e | LD e |
| 1 ORB | AND f | LD f |
| 2 ANB | ORB | ANB |
| AND g | ANB | ORB |
| 3 ORB | AND g | ANB |
| OUT h | ORB | LD g |
| | OUT h | ANB |
| | | ORB |
| | | OUT h |

Output Rung Examples

This section shows you examples of output rungs and their optimized instruction list representation. Like the input rungs, many output rungs have more than one possible instruction list representation. Therefore, where applicable an optional representation is shown to the right of the optimized list.

Single Output

It is possible to have a rung made up of just an output. An example of such a rung is shown below.

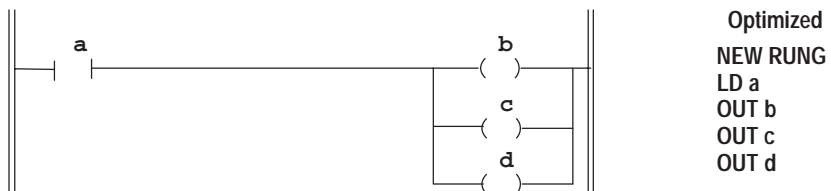


Optimized
NEW RUNG
OUT a

Output Branching

You can program parallel outputs on a rung to allow a true logic path to control multiple outputs, as shown in the following example.

Important: It is possible to have a rung with output branching and no input instruction.



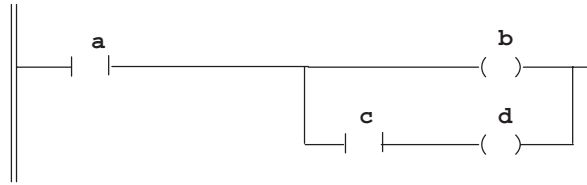
Optimized
NEW RUNG
LD a
OUT b
OUT c
OUT d

Output Branching With Conditions

As stated in chapter 6, additional input logic instructions, or conditions, can be programmed in the output branches to further control the outputs.

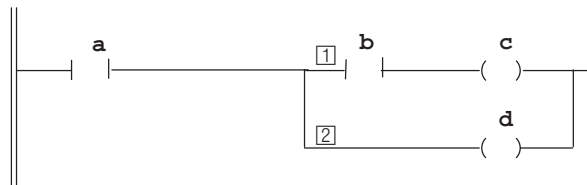
Four different examples of output branching with conditions are provided for you. Study each example to get a better understanding of when you need to use multiple output circuit connecting instructions (MPS, MRD, and MPP) in your output branches. (See page 8–10 for more information regarding the use of these instructions.)

Example 1 – Multiple output circuit that does not require connecting instructions.



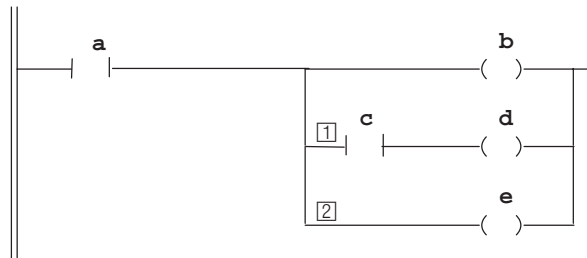
| Optimized | Option 1 |
|-----------|----------|
| NEW RUNG | NEW RUNG |
| LD a | LD a |
| OUT b | OUT b |
| AND c | LD c |
| OUT d | ANB |
| | OUT d |

Example 2 – This example requires MPS and MPP connecting instructions.



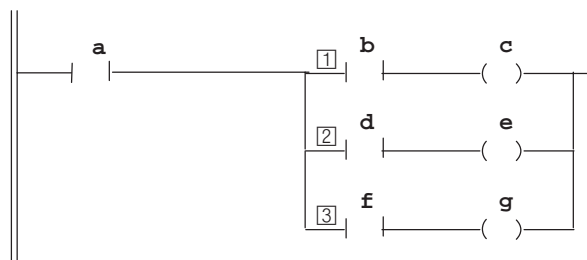
| Optimized | Option 1 |
|-----------|----------|
| NEW RUNG | NEW RUNG |
| LD a | LD a |
| ① MPS | MPS |
| AND b | LD b |
| OUT c | ANB |
| ② MPP | OUT c |
| OUT d | MPP |
| | OUT d |

Example 3 – This example also requires MPS and MPP instructions.



| Optimized | Option 1 |
|-----------|----------|
| NEW RUNG | NEW RUNG |
| LD a | LD a |
| OUT b | MPS |
| ① MPS | OUT b |
| AND c | MRD |
| OUT d | AND c |
| ② MPP | OUT d |
| OUT e | MPP |
| | OUT e |

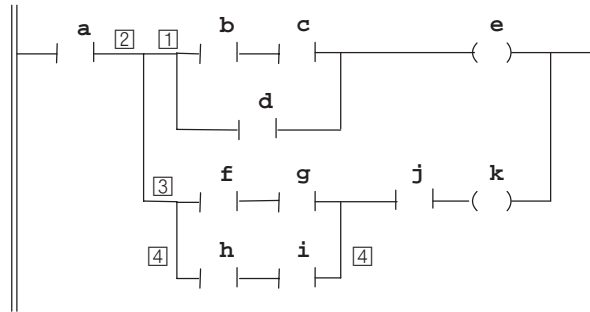
Example 4 – This example requires all three output circuit connectors: MPS, MRD, and MPP.



| Optimized | Option 1 |
|-----------|----------|
| NEW RUNG | NEW RUNG |
| LD a | LD a |
| ① MPS | MPS |
| AND b | LD b |
| OUT c | ANB |
| ② MRD | OUT c |
| AND d | MRD |
| OUT e | LD d |
| ③ MPP | ANB |
| AND f | OUT e |
| OUT g | MPP |
| | LD f |
| | ANB |
| | OUT g |

Output Branching with Block Connections

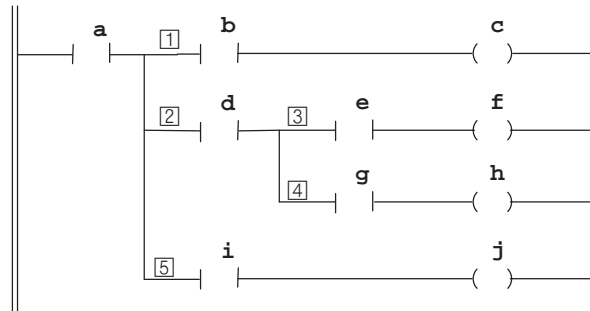
The example below shows how you use ANB and ORB instructions in a rung's output circuit. (See page 8–12 for more information regarding the use of these instructions.)



| | Optimized | Option 1 |
|---|-----------|----------|
| | NEW RUNG | NEW RUNG |
| | LD a | LD a |
| 1 | MPS | MPS |
| | LD b | LD b |
| | AND c | LD c |
| | OR d | ANB |
| 2 | ANB | LD d |
| | OUT e | ORB |
| 3 | MPP | ANB |
| | LD f | OUT e |
| | AND g | MPP |
| | LD h | LD f |
| | AND i | LD g |
| 4 | ORB | ANB |
| 2 | ANB | LD h |
| | AND j | LD i |
| | OUT k | ANB |
| | | ORB |
| | | ANB |
| | | AND j |
| | | OUT k |

Nested Output Branches

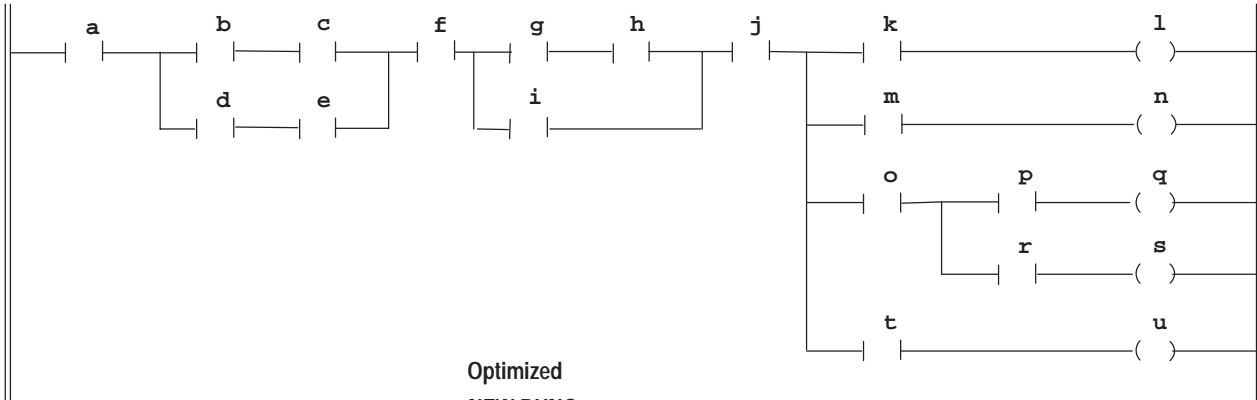
The example below requires multiple MPS/MPP instructions in a rung. (See page 8–10 for more information regarding the use of these instructions.)



| | Optimized | Option 1 |
|---|-----------|----------|
| | NEW RUNG | NEW RUNG |
| | LD a | LD a |
| 1 | MPS | MPS |
| | AND b | LD b |
| | OUT c | ANB |
| 2 | MRD | OUT c |
| | AND d | MRD |
| 3 | MPS | LD d |
| | AND e | ANB |
| | OUT f | MPS |
| 4 | MPP | LD e |
| | AND g | ANB |
| | OUT h | OUT f |
| 5 | MPP | MPP |
| | AND i | LD g |
| | OUT j | ANB |
| | | OUT h |
| | | MPP |
| | | LD i |
| | | ANB |
| | | OUT j |

Putting it All Together

The example that follows combines all of the input and output concepts that have been presented to you. The optimized instruction list is shown beneath the rung.



```

Optimized
NEW RUNG
LD a
LD b
AND c
LD d
AND e
ORB
ANB
AND f
LD g
AND h
OR i
ANB
AND j
MPS
AND k
OUT l
MRD
AND m
OUT n
MRD
AND o
MPS
AND p
OUT q
MPP
AND r
OUT s
MPP
AND t
OUT u
    
```

Programming Considerations

Since more than one instruction list representation may be possible for a single rung, you should be aware of the following programming issues if you are using programming software with your MicroLogix 1000 HHP:

- Any rung created in the HHP and saved to the controller, uploaded to programming software and edited, downloaded to the controller, and then monitored by the HHP will have the optimized instruction list representation. Thus, the program size and scan time may decrease if you originally created a program with a non-optimized list representation.
- Any rung created using programming software, downloaded to the controller, and then monitored by the HHP will have the optimized instruction list representation.
- Any rung created using programming software, downloaded to the controller, edited by the HHP and saved to the controller, and then uploaded to the programming software will have all unnecessary branches removed. Thus, the program size and scan time may decrease.

Entering and Editing Your Program

Read this chapter to enter and edit program files using your MicroLogix 1000 HHP. This chapter describes:

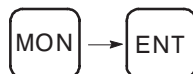
- entering the program monitor
- editing considerations
- editing in append and overwrite modes
- deleting instructions and rungs
- searching for specific addresses

Entering the Program Monitor

Once you understand the structure of instruction lists, you can begin entering the instructions into the program files. You do this in the program monitor functional area. This section shows you how to access the program monitor and describes the main screens within it.

Accessing the Program Monitor

To access the program monitor display for the program, begin at the home screen and press the key sequence shown here:



If this is the first time you are entering the program monitor, you will see the Start of File screen (explained below). If you entered the program monitor before, you return to the last location you were at within the program.

Program Monitor Screen Definitions

There are four key screens used within the program monitor. The Start of File and the End of File screens are created automatically as part of every program file. The Start of Rung and First Instruction on Rung screens are created by you as you add rungs to the program files. Each of these screens is explained below.

Start of File Screen

```
P  S T A R T   F I L E : 0 2
M A I N _ P R O G
```

The Start of File screen signifies the top of a program file. It is from this screen that you add the first rung to the file.

File names appear in the lower left-hand corner of only two of the Start of File screens. Program file 2 has the default name `MAIN_PROG` since it is the main program file, and file 3 has the default name `USER_FAULT` since this program file is only executed when a fault occurs. (The program file names can be changed using the programming software.)

End of File Screen

```
P 0 0 0   E N D
          F I L E : 0 2
```

As the name implies, this screen signifies the end of a program file. If you do not add any rungs to a program file, the End of File screen is labeled rung 0, as it is considered the only rung in the file.

Start of Rung Screen

```
P 0 0 0  ┆
```

The Start of Rung screen shows the Start of Rung (SOR) symbol. As the name implies, this symbol indicates the start of a new rung.

Important: Every rung *must* begin with an SOR symbol. To add an SOR symbol to the program file, press the key shown here:



To create the first rung in the program file, you must press this key while viewing the Start of File screen.

First Instruction on Rung Screen

```
P 0 0 0  ┆┆┆
I / 6                                     0
```

The First Instruction on Rung screen shows the SOR symbol to the left of the first instruction entered onto the rung. For example, the screen above shows the SOR symbol to the left of a LD instruction.

Editing Considerations

Before you begin to edit an existing program, you should store the program on a memory module (or save it on a personal computer with programming software). That way, if you edit a program and then decide that you don't want to accept the edits you have made (i.e., you want the original program back), you can reload the original unedited program to the controller.

Editing Modes

When the controller is in RPRG mode, you can edit program files from within the MicroLogix 1000 HHPs program monitor functional area. The HHP has two editing modes:

- append (P)
- overwrite (O)

A letter appears in the upper left-hand corner of the program monitor display to indicate what mode is currently selected. This letter starts flashing once edits are made to the program. The default mode is append (P).

Edit Mode



```

P  STARTFILE: 02
  MAIN_PROG
    
```

In general, append mode is used to add a new instruction or rung, while overwrite mode is used to write over or edit the parameters of existing instructions. To toggle between the edit modes, press the key shown here:

OVR

Each mode is explained in more detail below.

Append Mode

To add an instruction or a rung to a program, the MicroLogix 1000 HHP must be in append mode. The instruction or rung is always added *after* the instruction or rung currently displayed by the HHP.

Adding an Instruction

Follow these steps to add an instruction:

1. Make sure that the edit mode is set to append (P). Toggle the edit mode key if necessary.
2. Arrow to the location where you want to add an instruction. The table below describes from where you can add an instruction.

| You can add an instruction: | But you must: |
|---|--|
| at the beginning of an existing rung | go to the Start of Rung screen. |
| after an existing instruction on a rung | have the existing instruction displayed (any parameter of it). |

Important: You cannot add an instruction when the current display is the End of File screen. To add an instruction to the end of a rung, you must be on the rung's last instruction.

3. Press the instruction key or enter the function code corresponding to the instruction you want to enter.

Adding a Rung

Follow these steps to add a rung:

1. Make sure that the edit mode is set to append (P). Toggle the edit mode key if necessary.



2. Arrow to the location you want to add a rung. The table below describes from where you can add a rung.

| You can add a rung: | But you must: |
|------------------------------------|---|
| at the beginning of a program file | go to the Start of File screen. |
| after an existing rung | have the existing rung displayed (any instruction on it). |

Important: You cannot add a rung when the current display is the End of File screen. To add a rung to the end of the program file, you must be on the file's last rung.

3. Add the rung by pressing the New Rung key.



4. Add the instructions you want on the new rung.

Overwrite Mode

With the MicroLogix 1000 HHP in overwrite mode, you can either write over the current parameters of an instruction or you can write over an entire instruction to replace it with a new one.

Overwriting an Instruction's Parameters

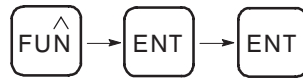
The steps below describe how to write over the parameters of an existing instruction.

1. Change the edit mode to overwrite (if you haven't already).



2. Arrow to the instruction with the parameters you want to change.

3. Press the key sequence shown below.



4. You are now able to change any or all of the current parameters for this instruction. The table below describes your options.

| If you want to: | Then you should: |
|-------------------------------|---|
| accept the current parameters | press ENT . |
| change the current parameters | use DEL as a destructive backspace, or FUN-DEL to delete the entire displayed address. Then key in the desired address and press ENT . |

Once the instruction is accepted, the instruction immediately after the instruction you wrote over is displayed.

Overwriting an Instruction

The steps below describe how to write over an existing instruction.

1. Change the edit mode to overwrite (if you haven't already).



2. Arrow to the instruction you want to replace.

Important: You cannot write over the Start of File, Start of Rung, or End of File screens.

3. Press the instruction key or enter the function code corresponding to the instruction you want to enter.

- A.** If this instruction is of the same instruction class as the instruction it is overwriting (e.g., LD and LDI), the current parameters are loaded for the new instruction. The table below describes your options.

| If you want to: | Then you should: |
|-------------------------------|---|
| accept the current parameters | press ENT . |
| change the current parameters | use DEL as a destructive backspace, or FUN-DEL to delete the entire displayed address. Then key in the desired address and press ENT . |

- B.** If this instruction is of a different instruction class, no parameters are loaded. Key in the desired address and press **ENT**.

Once the instruction is accepted, the instruction immediately after the instruction you wrote over is displayed.

Deleting Instructions and Rungs

While in either append or overwrite edit mode, you can delete individual instructions and rungs. You can also delete a range of rungs within a single program file.

Important: If you delete an instruction or rung(s), you *cannot* undo the deletion.

Deleting an Instruction

You can delete a single instruction while viewing any parameter of that instruction by pressing the key shown below.



Important: A confirmation screen will *not* appear; the instruction is simply deleted. If you delete an instruction, you *cannot* undo the deletion.

Once the instruction is deleted, the instruction that immediately followed it on the rung appears on the display. The only exceptions to that rule are described in the table below:

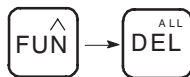
| If the deleted instruction was: | Then the display shows: |
|---------------------------------|---|
| the last instruction on a rung | the first instruction of the next rung. |
| the only instruction on a rung | an empty Start of Rung screen. |

You cannot delete the Start of Rung screen unless all other instructions on the rung are deleted first. However, you can delete the entire rung.

Deleting a Single Rung

To delete a single rung, follow these steps:

1. While viewing any instruction located on the rung you want to delete, press the key sequence shown here:



```

DELETED RUNG(S) :
X X X
    
```

The confirmation screen shown above appears. The number of the rung to be deleted is shown in the lower left-hand corner.

2. If this is the rung you want to delete, press **ENT**. If you do not want to delete this rung, press **ESC**.

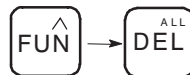
Important: If you delete a rung, you *cannot* undo the deletion.

Once the rung is deleted, the rung that immediately followed it in the program file is displayed.

Deleting a Range of Rungs

Deleting a range of rungs uses the same key sequence required to delete a single rung. Follow these steps to delete multiple rungs:

1. Start at the first or last rung of the range of rungs you want to delete.
2. While viewing any instruction located on that rung, press the key sequence shown here:



```
DELETERUNG(S) :  
XXX
```

The confirmation screen shown above appears. The number of the first rung to be deleted is shown in the lower left-hand corner.

3. Press the key shown below to indicate you are deleting a range of rungs.



A dash appears on the screen.

4. Type in the number of the last rung to be deleted. This number can be less than or greater than the first rung. The screen now looks like this:

```
DELETERUNG(S) :  
XXX - XXX
```

5. Press **ENT** to delete the range of rungs. The rungs indicated and all the rungs in between are deleted. To abort this procedure, press **ESC**.

Important: If you delete a range of rungs, you *cannot* undo the deletion.

Once the range of rungs is deleted, the rung that immediately followed it in the program file is displayed.

Searching for Specific Addresses

The search option allows you to quickly locate addresses in program files. You can search for either an address that you enter or an address that is displayed.

Searching for an Address You Enter

To enter an address and search for it, press the key sequence shown here:



This type of search can be invoked from any of the four functional areas:

- home screen
- program monitor
- data monitor
- multi-point

The search begins from the top of the first program file and searches down through all of the files. If the search option cannot find the address you enter, the message NOT FOUND is displayed.

Searching for an Address That is Displayed

You can search for the address that is currently displayed on the MicroLogix 1000 HHP by pressing the key shown here:



The table below shows the functional areas from which the search can be invoked and from where in the program the search begins.

| From this Functional Area: | The Search Begins From the: |
|----------------------------|--|
| Program Monitor | current display and searches down to the last program file. <i>Search does not wrap around from the last file to the first file.</i> |
| Data Monitor | top of the first program file and searches down through all of the files. |
| Multi-Point | |

If the search option cannot find the displayed address, the message NOT FOUND appears.

Searching for Bit Addresses Versus Word Addresses

The following table outlines whether a search finds a bit address or a word address when searching from the home screen, program monitor, or multi-point functional areas.

| If the address entered or displayed is a: | Then the search finds: |
|---|---|
| bit address | only bit instructions referencing that bit. |
| word address | only word instructions referencing that word. |

The table below shows whether a search finds a bit address or a word address when searching from the data monitor functional area.

| If the address entered or displayed is a(n): | Then the search finds: |
|--|---|
| output, input, bit, or integer address | either word or bit instructions, depending on the current radix: integers and hexadecimal radices find word addresses, while binary radices find bit addresses. |
| timer, counter, or control address | any bit instructions or parameters of word instructions referencing that Timer, Counter, or Control element. |
| status file address | either word or bit instructions, depending on the level currently displayed. |

After You've Entered Your Program

This chapter shows you the procedures for:

- changing the program configuration defaults
- accepting your program edits
- changing processor modes
- monitoring your controller
- viewing data table files
- using the multi-point function
- forcing inputs and outputs

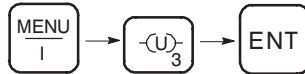
Changing the Program Configuration Defaults

Every default MICRO program is configured with the settings shown in the table below. The table shows the default program configuration settings and indicates whether you need to save your program for the configuration changes to take affect.

| Program Configuration | Default Setting | Must Save Program for Change to Take Affect | See Page |
|--|---|---|----------|
| Program Name | MICRO | Yes | 18-2 |
| User Password | No password | No | 18-2 |
| Master Password | No password | No | 18-2 |
| Run Always | No | Yes | 18-5 |
| Start-Up Protection | No | Yes | 18-6 |
| Fault Override | No | Yes | 18-7 |
| Extended I/O Configuration ^① | No | Yes | 18-8 |
| STI Setpoint | 0 (x 10 ms) | No | 18-9 |
| STI Enabled | Yes | Yes | 18-10 |
| Watchdog Scan | 10 (x 10 ms) | No | 18-11 |
| Input Filters | 8.00 msec for all groups | Yes | 18-12 |
| Analog Config Channel Enable Filter Setting Output Config | All Channels Enabled 60 Hz Voltage Output | Yes | 18-14 |
| Lock Program | No | Yes | 18-17 |
| Controller Version | Series A and B discrete controllers | Yes | 18-18 |

^① Valid for Series A – C discrete only.

You can change these default settings by accessing the program configuration menu:



Naming the Program

You can enter a new name for your program using any combination of the letters or numbers available on the MicroLogix 1000 HHP keypad (i.e., A–F,I,N,O,R,S,T, and 0–9). The maximum length is eight characters.

Follow the steps below to change the program name.

1. Access the program configuration menu.
2. Select the first option, NAME PROG.



3. Type in a new name for your program.
4. Enter the name and return to the previous functional area.



Your new program name is now displayed in the upper left-hand corner.

5. You must accept your program edits for this change to take affect. For information on accepting edits, see page 18–21.

Setting the User and Master Passwords

Password protection prevents access to a program and prevents changes from being made to the program. Each program may contain two passwords: the *user password* and the *master password*. The master password overrides the user password.

You can use passwords in the following combinations:

| | |
|--|--|
| Only User Password Designated | You must enter the user password to gain access to the program. |
| Only Master Password Designated | You do not have to enter the master password to gain access to the program. A master password is used by itself to allow access if a user password has been entered. |
| User Password and Master Password Designated | You must enter either the user password or the master password to gain access to the program. |

Generally, if you are using a number of controllers, each controller is given a different user password and a master password is applied to all of the controllers. You can use the master password to change or remove any password.

Important: There is no *password override* to defeat the protection. Contact your Allen-Bradley representative if you are not able to locate your password.

To enter a user password and/or master password:

1. Access the program configuration menu.
2. Arrow down to the option `USER PASSWORD` or `MASTER PASSWORD` and select it.



N E W P A S S W R D :

Important: If a password already exists, you are prompted to enter it before you can enter a new password.

3. Type in a new password using any combination of the numbers available on the MicroLogix 1000 HHP keypad (i.e., 0–9). You can only use numeric-based passwords. The maximum length for a password is ten characters.
4. Enter the new password.



Another screen appears so you can verify the new password.

5. Retype the new password, enter it, and return to the home screen.



Important: To remove password protection from a program, you need to enter an empty password. To do this, follow the steps shown above but do not type (or retype) a new password when prompted. Instead, just press **ENT** to bypass the prompts.

Setting the Run Always Bit

This selection determines what mode the controller will enter at power up following a power down or an unexpected reset.

When this bit is set to **NO**, the controller powers up in the mode it was in prior to losing power, with one exception: If the controller was in one of the test modes (RCSN or RSSN) when power was removed, the controller enters RPRG when power is applied. With this setting you must manually clear the faults that caused the power down.

When this bit is set to **YES**, the controller automatically clears any major and minor errors that caused the power down or unexpected reset and then attempts to enter RRUN mode when power is applied.



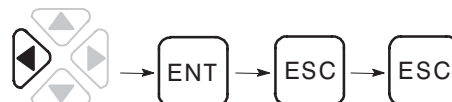
ATTENTION: Unexpected resets may occur due to electromagnetic noise, improper grounding, or an internal controller hardware failure. If an unexpected reset occurs and the Run Always bit is set, the controller will enter the RRUN mode; therefore, make sure your application is designed to safely handle this situation.

To change the bit setting:

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Access the program configuration menu.
3. Arrow down to the option **RUN ALWAYS** and select it.



4. Select the option **YES** and return to the home screen.



5. You must accept your program edits for this change to take affect. For information on accepting edits, see page 18–21.

Setting the Start-Up Protection Bit

This selection allows you to check for, and attempt to recover from, major errors before starting the first scan of your program.

When this bit is set to **NO**, the controller will fault if a major error occurs while in **RRUN**, **RCSN**, or **RSSN** mode.

When this bit is set to **YES** and power is cycled when the controller starts in **RRUN** mode, the controller executes the user-fault routine prior to the execution of the first program scan. You can then clear the Major Error Halted bit (S1/13) to resume operation in the **RRUN** mode.

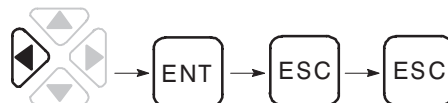
If user-fault routine does not reset bit S1/13, the fault mode results. Therefore, program the user-fault routine logic accordingly. When executing the startup protection fault routine, the Major Error Fault (S6) will contain the value 0016H.

To change the bit setting:

1. Put the controller in **RPRG** mode (if it is not already in that mode).
2. Access the program configuration menu.
3. Arrow down to the option **START PROTECT** and select it.



4. Select the option **YES** and return to the home screen.



5. You must accept your program edits for this change to take affect. For information on accepting edits, see page 18–21.

Setting the Fault Override Bit

If the controller is faulted while in RRUN mode, this selection determines whether or not the controller attempts to clear the errors at power up.

When this bit is set to NO, after power is cycled in the controller, you must manually clear the faults that occurred while in RRUN mode.

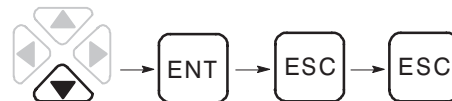
When this bit is set to YES, the controller automatically clears any major and minor errors on power up and then attempts to enter RRUN mode when power is applied.

To change the bit setting:

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Access the program configuration menu.
3. Arrow down to the option `FLT OVERRIDE` and select it.



4. Select the option YES and return to the home screen.



5. You must accept your program edits for this change to take affect. For information on accepting edits, see page 18–21.

Setting the Extended I/O Configuration Bit

When this bit is set to **NO** and unused outputs are written to, the controller will fault. Therefore, this bit must be set to **YES** if writing to unused outputs. (For a 16 I/O controller, O/6–O/15 are unused outputs. For a 32 I/O controller, O/12–O/15 are unused outputs.)

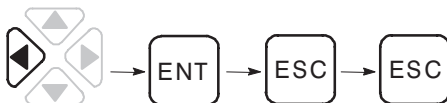
Note: This selection is valid for Series A–C discrete only.

To change the bit setting:

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Access the program configuration menu.
3. Arrow down to the option **EXTEND IO CNFG** and select it.



4. Select the option **YES** and return to the home screen.



5. You must accept your program edits for this change to take affect. For information on accepting edits, see page 18–21.

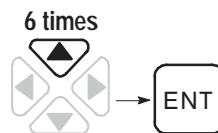
Setting the STI Setpoint

The STI setpoint is the time between successive executions of the STI subroutine. The allowable range is from 10 ms to 2550 ms (entered in 10 ms increments). A setpoint of zero disables the setpoint function.

Important: The setpoint value must be a longer time than the execution time of the STI subroutine file, or a minor error bit is set.

To change the setting:

1. Access the program configuration menu.
2. Arrow up to the option STI SETPOINT and select it.



3. Type in a new setpoint.
4. Enter the setpoint and return to the home screen.



Setting the STI Enabled Bit

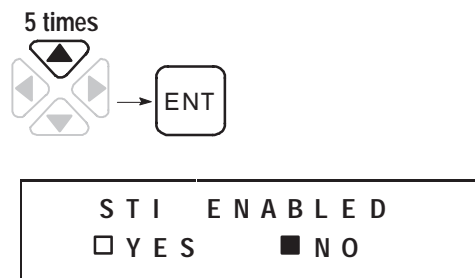
This selection determines if execution of the STI is allowed.

When this bit is set to **NO**, execution of the STI is not allowed.

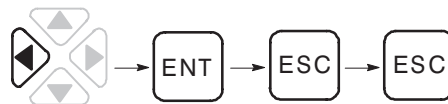
When this bit is set to **YES**, execution of the STI is allowed (provided the STI setpoint is non-zero). If reset when an interrupt occurs, the STI subroutine does not execute and the STI Pending bit is set. The STI Timer continues to run when this bit is disabled. The STE instruction sets this bit. The STD instruction clears this bit.

To change the bit setting:

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Access the program configuration menu.
3. Arrow up to the option **STI ENABLED** and select it.



4. Select the option **YES** and return to the home screen.



5. You must accept your program edits for this change to take affect. For information on accepting edits, see page 18–21.

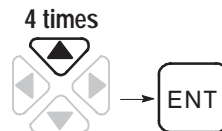
Selecting the Watchdog Scan

This byte value contains the number of 10 ms ticks allowed during a program cycle. The default value is 10 (100 ms), but you can increase this to 255 (2.55 seconds) or decrease it to 1, as your application requires.

Important: If the watchdog value equals the current scan time value, a watchdog major error will be declared (code 0022).

To change the setting:

1. Access the program configuration menu.
2. Arrow up to the option WATCHDOG SCAN and select it.



3. Type in a new scan time.
4. Enter the setpoint and return to the home screen.



Setting the Input Filters

This option allows you to select input filter response times for the 1761-L16BWA and 1761-L32BWA micro controllers.

The input filter response time is the time from when the external input voltage reaches an on or off state to when the micro controller recognizes that change of state. The higher you set the response time, the longer it takes for the input state change to reach the micro controller. However, setting higher response times also provides better filtering of high frequency noise.



You can apply a unique input filter setting to each of the three input groups:

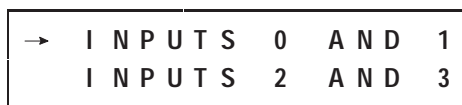
- 0 and 1
- 2 and 3
- 4 to x; where x=5 for 10 I/O point controllers, x=9 for 16 I/O point controllers, x=19 for 32 I/O point controllers, and x=11 for analog controllers

Important: The input filter response times for the 1761-L16AWA and 1761-L32AWA micro controllers are fixed at 8ms. Selecting any other input filter response time for these controllers will not change the response time of the inputs. However, your response times given by your software will *not* show the actual times. Also, the Input Filter Modified Bit (S5/13) will be set when the controller is in run or test mode.

The minimum and maximum response times associated with each input filter setting can be found on page A-6.

To change the input filter settings:

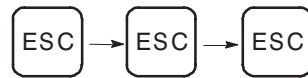
1. Put the controller in RPRG mode (if it is not already in that mode).
2. Access the program configuration menu.
3. Arrow up to the option INPUT FILTERS and select it.



4. Scroll up or down to the input group you want to change and select it. The following display appears:

| | | |
|---|-----------|---------|
| → | 8 . 0 0 | M S E C |
| | 1 6 . 0 0 | M S E C |

5. Scroll to the response time you want and select it.
6. Repeat steps 3 and 4 as needed, then return to the home screen.



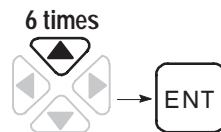
7. You must accept your program edits for this change to take affect. For information on accepting edits, see page 18–21.

Selecting the Filter Setting for Analog Input Channels

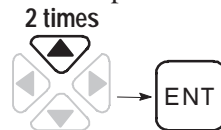
This option allows you to select the filter setting for the analog input channels. The same setting is used for all four channels.

To change the filter setting for analog input channels:

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Access the program configuration menu.



3. Arrow up to the option `ANALOG CONFIG` and select it.



Note: The function is only available when the Micro Term is attached to an analog controller.



4. Arrow up to the option `FILTER SETTING` and select it.



Select a filter setting from the following list:

- 10 Hz
- 50 Hz
- 60 Hz
- 250 Hz

5. Scroll to the setting you want and select it.

6. Enter to store the filter setting and return to the `ANALOG CONFIG` submenu.



7. Escape to return to the submenu *without* changing the filter setting.



Selecting Channel Enable for the Analog Input Channels

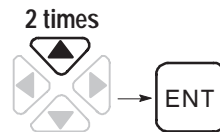
This option allows you to enable/disable input channels 0–3 of the analog controller.

To enable/disable the analog input channels:

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Access the program configuration menu.



3. Arrow up to the option `ANALOG CONFIG` and select it.



Note: The function is only available when the Micro Term is attached to an analog controller.



4. Enter to select the option `CHANNEL ENABLE`.



5. Arrow left or right to the setting you want for the displayed channel (enabled or disabled) and select it.



6. Enter to store the selected setting and advance to the next channel screen.



7. Scroll up/down to move to the previous/next channel screen *without* storing the setting.

8. Escape to return to the `ANALOG CONFIG` submenu *without* storing the current setting.

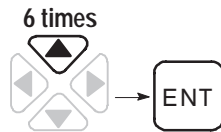


Configuring the Analog Output

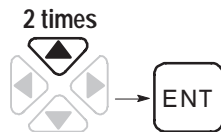
The analog controller's output channel can be configured to support either current *or* voltage operation.

To configure the analog output:

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Access the program configuration menu.



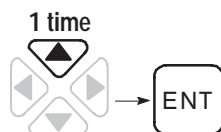
3. Arrow up to the option ANALOG CONFIG and select it.



Note: The function is only available when the Micro Term is attached to an analog controller.



4. Arrow up to the option OUTPUT CONFIG and select it.



5. Arrow up/down to move between the two settings (0 – 10V or 4 – 20 mA).



6. Enter to store the selected setting and return to the ANALOG CONFIG submenu.



7. Escape to return to the to ANALOG CONFIG submenu *without* storing the current setting.



Setting the Lock Program Function

This option allows you to prevent proprietary algorithms from being viewed on the display or from being stored in a memory module. When this function is set to NO, access to a controller program is unrestricted. This is the default.

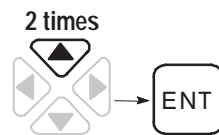
When this function is set to YES, access to a controller program is only permitted when a matching program exists in the memory module. If a memory module is not present or a matching program does not exist, you can only perform the following functions:

- change the language
- change the display contrast
- change the controller mode
- view/clear faults
- monitor/edit data
- view the multi-point list
- force I/O and/or clear forces
- clear the controller program
- clear a memory module program
- load a different program

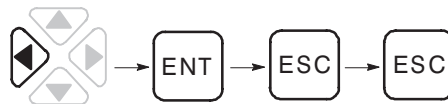
Important: If you lose or delete the memory module's copy of the program, you must clear the controller memory and re-enter the program.

To change the function setting:

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Access the program configuration menu.
3. Arrow up to the option `LOCK PROG` and select it.



4. Select the option `YES` and return to the home screen.



5. You must accept your program edits for this change to take affect. For information on accepting edits, see the section that follows.

Important: You must cycle power on the MicroLogix 1000 HHP for the lock to be enforced.

Changing the Version of Controllers that the Program Supports

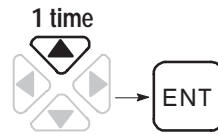
The `CONTROL VERSION` option allows your program to access the communication functions of Series C/D MicroLogix 1000 discrete controllers.

Important: The control version function is only available when the Micro Term is attached to a series C/D discrete controller containing a series A/B program. It is not available when the Micro Term is attached to an analog controller.

To change the default setting (`ML-A/B`):

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Access the program configuration menu.

3. Arrow up to the option `CONTROL VERSION` and select it.

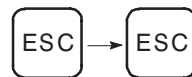


| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | E | T | T | Y | P | E | | M | L | - | C | / | D | |
| Y | E | S | [| E | N | T |] | N | O | [| E | S | C |] |

4. Press `ENT` to select support of Series C/D discrete controllers. (Press `ESC` to continue support of Series A and B discrete controllers.)

Important: You cannot revert a program back to Series A or B discrete controller support once you've configured it for Series C/D discrete controller support.

5. Return to the home screen.



6. You must accept your program edits for this change to take affect. For information on accepting edits, see page 18–21.

Accepting Your Program Edits

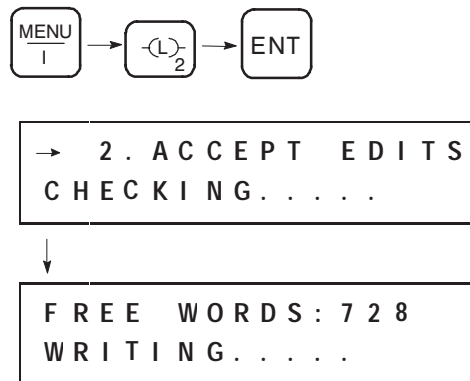
There are two ways you can accept the edits made to your program. Both of these methods initiate the verification and save of the program. These methods are:

- using the `ACCEPT EDITS` option from the menu while in RPRG mode.
- changing from RPRG mode to any run or test mode (RRUN, RCSN, or RSSN)

Using the `ACCEPT EDITS` Menu Option

To use this method, select `ACCEPT EDITS` from the menu as shown below.

Important: A confirmation screen does not appear. Be certain you want to accept the edits you've made to the program before you select this menu option.



When the verification is completed, the screen displays the number of free user words. If an error message appears, see chapter 19 for definitions of the program verification error codes and the recommended action(s) you should take.

Once the program edits are accepted, the menu screen reappears. The controller remains in RPRG mode.

Changing from RPRG Mode to any Run or Test Mode

For information on accepting edits using this method, see the section below, Changing Controller Modes.

Changing Controller Modes

The following section describes the available modes of operation for the micro controller, how to change between those modes using the MicroLogix 1000 HHP, and the tasks you can perform in each mode.

Modes of Operation

All controller modes are considered to be remote, which indicates that the modes can be changed via the communication channels.

The current mode of the controller is shown in the upper right-hand corner of the MicroLogix 1000 HHPs home screen. The table below shows the possible display entries and the corresponding micro controller mode.

| Display Entry | Micro Controller Mode |
|---------------|-------------------------------|
| RPRG | Remote Program |
| RRUN | Remote Run |
| RCSN | Remote Test – Continuous Scan |
| RSSN | Remote Test – Single Scan |
| RSUS | Remote Suspend ^① |
| FLT | Fault ^② |

① The controller only enters suspend mode if you run a program that executes a suspend instruction.

② The controller only enters fault mode if, while a program is executing, a fault occurs within the operating system or the program, or setting S1/13 in any mode. See page 20–11 for information on identifying and clearing faults.

You can place the micro controller in remote program, run, and test modes using the mode key. Each of these modes is described below. (Changing modes is described on page 18–22.)

Remote Program Mode

The remote program mode (RPRG) allows you to modify your program, edit data, and transfer programs to and from the memory module. In this mode the processor does not scan or execute the ladder program and all outputs are de-energized.

Remote Test Mode

While you are in the remote test mode, the controller monitors input devices, scans or executes the program, and updates the output data files without energizing output circuits or devices. There are two remote test modes available:

- **Remote Continuous Scan (RCSN)** – This test mode is the same as the remote run mode (described below), except output circuits are not energized. This allows you to troubleshoot or test your program without energizing external output devices on a continuous basis.
- **Remote Single Scan (RSSN)** – In this test mode, the controller executes a single operating cycle that reads the inputs, executes the program, and updates all data without energizing output circuits.

Remote Run Mode

While you are in the remote run mode (RRUN), the controller monitors input devices, scans or executes the program, energizes output devices, and acts on enabled I/O forces.

Changing Remote Modes

The following steps show how to change from RPRG mode to RRUN mode.

Important: When changing from RPRG mode to any other mode of operation (RRUN, RCSN, or RSSN), any edits that exist in the current program are accepted automatically. (Edits exist whenever the mode is flashing on the HHP display.)

1. Enter the mode options.



```

ACTIVE MODE: R P R G
←  ■ R P R G   □ R R U N  →
  
```

The current mode, RPRG, is shown on the top line. The bottom line shows the mode options. The arrows to the left and right indicate that more options are available.

2. Arrow over to RRUN.



```

ACTIVE MODE: R P R G
←  □ R P R G   ■ R R U N  →
  
```

3. Select RRUN. If edits exist in the current program, the program is checked and if accepted, the home screen appears. If you get a fault code, refer to chapter 19 to clear the fault. If an error message appears, see chapter 19 for a definition of the program verification error code and the recommended action(s) you should take.



```

M I C R O                R R U N
F R E E : 7 2 8   F I L E : 0 2
  
```

RRUN now appears in the upper right-hand corner of the screen. Also, the number of free instruction words is displayed.

Tasks You Can Perform

This table shows you what tasks you are allowed to perform in each of the possible modes.

| Activity | Remote Controller Mode | | | | |
|--|------------------------|------|-----|-------|---------|
| | Program | Test | Run | Fault | Suspend |
| Change the language. | X | X | X | X | X |
| Accept program edits. | X | | | | |
| Change the following program configuration settings: <ul style="list-style-type: none"> • Program Name • Run Always • Start-Up Protection • Fault Override • Extend I/O Configuration • STI Enabled • Input Filters • Lock Program | X | | | | |
| Change the following program configuration settings: <ul style="list-style-type: none"> • User Password • Master Password • STI Setpoint • Watchdog Scan | X | X | X | X | X |
| Load and store programs using the memory module. | X | | | | |
| Clear programs in the memory module. | X | X | X | X | X |
| Clear all forces. | X | X | X | X | X |
| Clear a program in the controller. | X | | | X | X |
| Change the baud rate and contrast. | X | X | X | X | X |
| Change to a different mode. | X | X | X | | X |
| Run your program with output points disabled. | | X | | | |
| View faults manually. | X | X | X | X | X |
| Force inputs and outputs. | X | X | X | X | X |
| Monitor and edit the program in the controller (without rung state indication). | X | | | | |
| Monitor the program, rung state, and data as it is being executed. | | X | X | | |
| Monitor and edit data files. | X | X | X | X | X |
| Change the radix. | X | X | X | X | X |
| Use the search, trace, and multi-point functions. | X | X | X | X | X |
| Access the function code table. | X | X | X | X | X |

Monitoring Your Controller

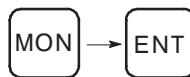
This section shows you how to monitor the program files and data table files. If a fault occurs while monitoring your controller, follow the procedure on page 20–11 to clear the fault.

Monitoring Program Files

Program files contain controller information, the main instruction list program, and any subroutine programs. Monitoring your program files allows you to watch the instruction parameters of timers, counters, and accumulated values change, and to see bits turn on and off as your program runs.

To monitor a program file, follow these steps:



1. From the home screen or data monitor, access the program monitor display for the program.



```
R  S T A R T  F I L E : 0 2
M I C R O
```

You return to the last location you were at within the program.

2. Arrow to the program file you want to monitor and move through the rungs, as described in the table below:

| To: | Press: |
|--|---|
| move up and down through a program's rungs and program files |  |
| move left and right through each rung of a program. (When the end of a rung is reached the next rung automatically scrolls into view as you move the cursor right or left in the program.) |  |

Using Short Cut Keys

The following table shows the short cut keys you can press to go directly to the file and rung you want to monitor.

| To go to: | Press the following key sequence: |
|---|--|
| a designated file and rung (e.g., 5/3) | MON → file number → $\frac{\text{ORB}}{+/- /}$ → rung number → ENT |
| rung 0 of a designated file (e.g., 10/0) | MON → file number → $\frac{\text{ORB}}{+/- /}$ → ENT |
| a designated rung in the current file (e.g., 4) | MON → rung number → ENT |

Monitoring Data Table Files

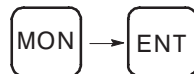
The data table files contain information used in your program. By monitoring the data table files, you can watch how the data changes as your program runs. Data table files that you can monitor include:

- Output (O)
- Input (I)
- Status (S)
- Bit (B)
- Timer (T)
- Counter (C)
- Control (R)
- Integer (N)



See page 18–27 for example displays of each data table file.

To monitor a data element used as an instruction parameter, begin at the program monitor display.

1. Arrow to the instruction parameter you want to monitor.
2. Access the data monitor to view the data table for the current instruction.

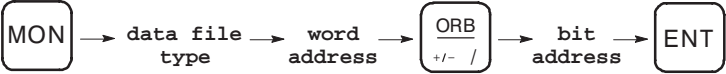
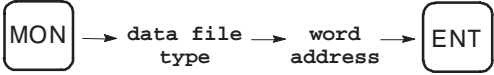
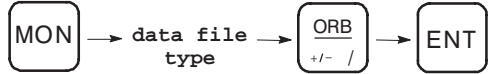
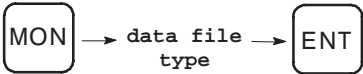


3. Scroll through the bits of individual data files or through the data file table using the keys described in the table below.

| To: | Press: |
|--|---|
| scroll through the bits of individual data files |  |
| scroll through the data file table |  |

Using Short Cut Keys

The following table shows the short cut keys you can press to go directly to the particular data file type, word address, and bit address you want to monitor.

| To go to: | Press the following key sequence: |
|---|--|
| a designated data file type, word, and bit (e.g., N5/3) |  |
| a designated data file type and word (e.g., N20) |  |
| word 0, bit 0 of a designated data file type (e.g., N0/0) |  |
| word 0 of a designated data file type (e.g., N0) |  |

Viewing Data Table Files

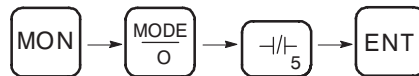
This section shows you how to access each type of data table file. It also describes how to change the radix display for output, input, bit, and integer data files.

Accessing Data Table Files

An example of how to access each type of data table file is shown below. The sample screen is provided to show you how each data table file looks.

Output Data Files

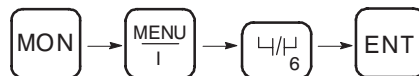
To access the output data file for O/5, press the key sequence shown below. For the output file type, the / character is automatically displayed by the MicroLogix 1000 HHP.



| | |
|---------------------|-------------|
| O / 5 | 0 0 |
| 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 1 0 |

Input Data Files

Access the input data file for I/6 as shown below. For the input file type, the / character is automatically displayed by the MicroLogix 1000 HHP.



| | |
|---------------------|---------------|
| I / 6 | I 0 |
| 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 |

Status Data File

To access the math register:



| | |
|------|-----------|
| S 13 | MATH REGL |
| | 0 0 0 0 H |

Bit Data File

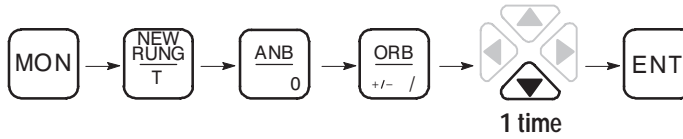
Access the bit data file for B/8 as follows:



| | |
|-----------------|-------------------|
| B / 8 | B 0 |
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 1 0 |

Timer Data File

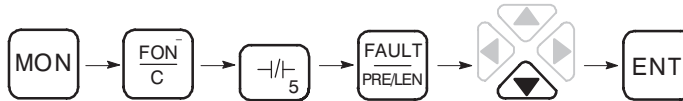
To view the Timer Enable Bit (EN) for T0:



| | | | |
|-------------|------|---------|---|
| T 0 0 | EN 1 | TT 1 | → |
| P 1 0 0 0 0 | A | 8 9 4 6 | |

Counter Data File

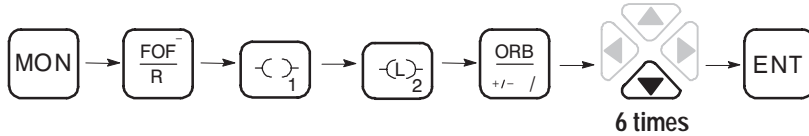
To access the Accumulator Value (ACC) for C5:



| | | | | |
|-------|---|-------|-------|-----|
| C 0 5 | ← | C D 0 | D N 0 | → |
| P | | 1 | A | - 1 |

Control Data File

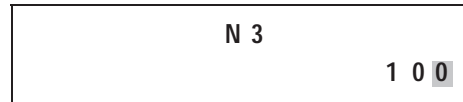
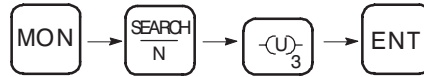
Access the Enable Bit (EN) for R12 as follows:



| | | | | |
|-------|---|------|-------|---|
| R 1 2 | ← | EN 0 | E U 0 | → |
| L | | 0 | P | 0 |

Integer Data File

To access integer data file N3:



Changing the Radix

Radix refers to the way numeric-based information is displayed. You can change the radix for output, input, bit, and integer data files. Since the displays for timer, counter, control, and status data files are pre-formatted, the option of changing the radix is not available for those data files.

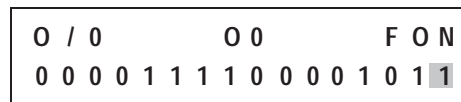
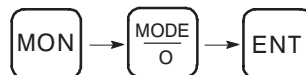
The radix choices are:

- binary (default radix for output, input, and bit files)
- hexadecimal
- decimal (default radix for integer file)

Important: When you exit the data monitor, the display remains in the last radix you selected.

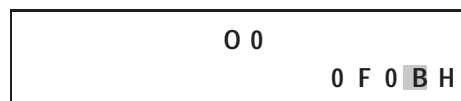
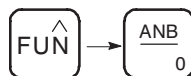
In the following example the display of an output file is changed from binary to hexadecimal to decimal and back to binary.

1. Access the output data table file.

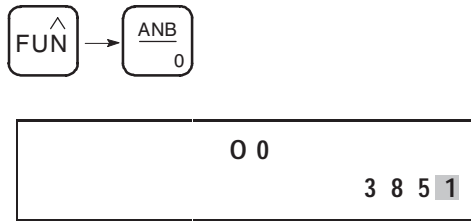


The output file defaults to a binary display.

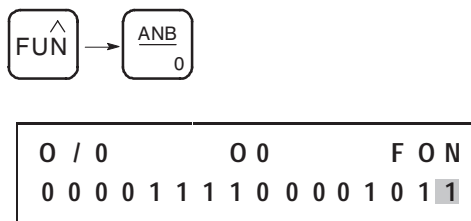
2. Change the display to a hexadecimal radix using the hot key sequence.



3. Change the display to a decimal radix.



4. Return to the binary radix display.



Important: Notice that the forces only appear in the binary radix.

Using the Multi-Point Function

The function allows you to monitor up to 16 non-contiguous bits of data at a time. Since the multi-point list is stored with the program, you can create a unique list for each micro controller program you create. The multi-point list stays with the program even if it is edited by programming software.

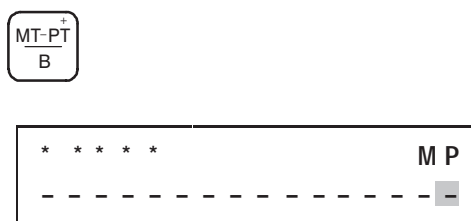
There are two ways to add addresses to the multi-point list: manually, entering each address while in the multi-point screen; or automatically, using short-cut keys from the program monitor or data monitor screens.

You can also change or delete addresses that are already entered into the multi-point list.

Manually Entering Addresses

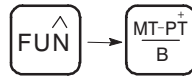
Follow the steps below to manually enter an address into the multi-point list.

1. Access the multi-point functional area.

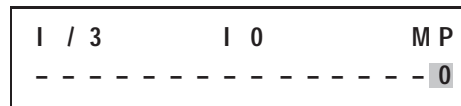
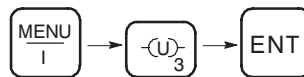


The dashed lines represent open points on the list, where addresses can be entered. If some addresses have already been entered to your program's list, the display will have 0s or 1s in place of some or all of the dashed lines.

2. Arrow to the position you want to enter an address. This can be any open point on the list.
3. Bring up a prompt to enter the address.



4. Enter the address of the bit you want to monitor. I/3 is shown here as an example:

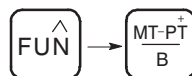


The address is added to the multi-point list at the location you selected in step 2.

Automatically Entering Addresses

You can use short-cut keys to add a bit address to the multi-point list. Follow the steps below.

1. While in either the program monitor or the data monitor, arrow to the bit address you want to add to the multi-point list.
2. Add the address to the list.



The address is added to the next available open point in the multi-point functional area.

Important: If the list is full, the message `LIST FULL` is displayed. The address is not added to the multi-point list.

3. If you want to verify that the address was added, enter the multi-point functional area.

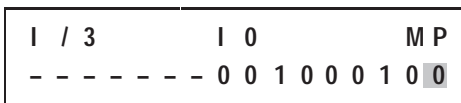


Press `ESC` to return to the last screen.

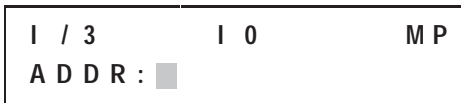
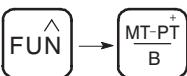
Changing Multi-Point Addresses

You can change an entry in the multi-point list by writing over the current address. Follow the steps below.

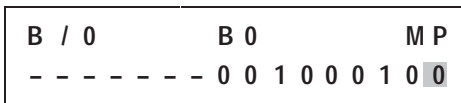
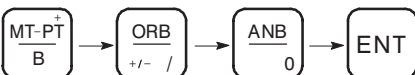
1. Access the multi-point functional area.



2. Arrow to the address you want to write over.
3. Bring up a prompt to enter the address.



4. Enter the address of the new bit you want to monitor. B/0 is shown here as an example:



The new address is added to the multi-point list at the location you selected in step 2.

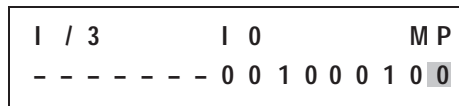
Removing Multi-Point Addresses

You can remove individual addresses from the multi-point list or you can delete the entire list.

Deleting Individual Addresses

Follow these steps to delete a single bit address from the multi-point list:

1. Access the multi-point functional area.



2. Arrow to the address you want to delete.
3. Remove the address from the list.

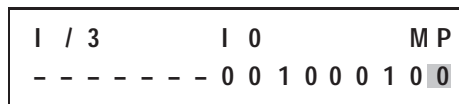


A dashed line replaces the bit data, indicating that this is now an open point on the list.

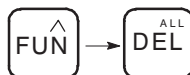
Deleting All Addresses

Follow these steps to remove all of the addresses currently entered in the multi-point list:

1. Access the multi-point functional area.



2. From any location in the list, press the delete all key sequence.



3. Confirm the deletion to remove the addresses.



Press **ESC** if you do not want to clear the list.

Forcing Inputs and Outputs

The Force function allows you to override the actual status of external input circuits by forcing external input data file bits On or Off. You can also override the controller logic and status of output data file bits by forcing output circuits On or Off.

Important: Forces are always enabled but must be set to be active.

You can set forces while the controller is in any mode.



ATTENTION: To avoid possible personal injury and damage to equipment, investigate the effects on machine operation before forcing external input data file bits or external output circuits.



ATTENTION: Forces override any output control from either the high-speed counter or from the output image. Forces may also be applied to the high-speed counter inputs. Forced inputs are recognized by the high-speed counter (e.g. forcing a count input On and Off increments the high-speed accumulator).

Forcing External Input Data File Bits

Setting forces on input data file bits affects the input data file and also the program logic. The effects on the program logic of set forces can be seen from the program monitor and the data monitor functional areas when in RRUN, RCSN, or RSSN.

The following is an example program monitor display of instruction I/6 in rung 0. The display indicates that the controller is in RRUN and no forces exist on this bit.

| | | | |
|---------|----|----|---|
| R 0 0 0 | ┆┆ | ┆□ | |
| I / 6 | | | 0 |

To force On the external input file address, follow the steps below.

1. Press the force On key. A confirmation screen appears.



2. Accept the confirmation to force the bit On.



This simulates the closing of the external input circuit. However, the actual open/closed status of the external input circuit no longer affects the program logic.

For the previous example, the following is now true:

- The instruction state box is filled, indicating that I/6 is true.
- FOn appears in the upper right-hand corner, indicating that the input is forced On.
- The data value, shown in the lower right-hand corner, is now a 1.
- The controller's Forced I/O LED is on continuously.

Guide to Forcing External Input Data File Bits

The following occurs in the Run or Test mode:

- Forcing of input data file bits and resultant data changes appear in the data file displays.
- The ladder program is scanned and ladder logic is applied.
- Instruction state boxes are filled for true bit instructions.

The table below shows the keys and key sequences involved with setting and clearing forces. A confirmation screen appears following each of these.

| Key or Key Sequence | Operation |
|---------------------|--|
| | If the controller is in RRUN, RCSN, or RSSN, the bit is forced Off and the data file bit remains forced until the force is removed. |
| | If the controller is in RRUN, RCSN, or RSSN, the bit is forced On and the data file bit remains forced until the force is removed. |
| or | Affects the cursored external input address. It removes the set force from the data file, if applicable. Other forces are unaffected. |
| | Affects all forced external input bit addresses and external output circuits. It removes set forces from all external input bit addresses and output circuits. |

The displays below show an example of setting a force while in Run or Test mode.

| | | | |
|---------|--|---|---|
| R 0 0 0 | | □ | 0 |
| I / 6 | | | |

Initial conditions.
Bit is Off.
No forces exist.

| | | | |
|---------|--|---|---|
| R 0 0 0 | | ■ | 1 |
| I / 6 | | | |

Bit is On.
No forces exist.

| | | | |
|---------|--|---|-------|
| R 0 0 0 | | ■ | F O N |
| I / 6 | | | 1 |

Bit is On.
Force is On.

| | | | |
|---------|--|---|-------|
| R 0 0 0 | | □ | F O F |
| I / 6 | | | 0 |

Bit is Off.
Force is Off.

Forcing an External Output Circuit

A forced external output circuit is independent of the internal logic of the program and the output data file. Setting forces on output circuits affects *only* the output circuit. Set forces do not affect the output data file or the program logic. The effects of set forces can be seen from the program monitor and the data monitor functional areas only while in RRUN. (RCSN and RSSN do not energize output circuits.)

The following is an example program monitor display of instruction O/1 in rung 1. The display indicates that the controller is in RRUN and no forces exist.

```
R 0 0 1    -( )- □
O / 1                                0
```

The procedure for forcing external output circuits is the same as forcing external input data file bits.

1. Press the force On key. A confirmation screen appears.

FON
C

```
FORCE BIT ON?
YES [ENT] NO [ESC]
```

2. Accept the confirmation to force the bit On.

ENT

```
R 0 0 1    -( )- □    F O N
O / 1                                0
```

For the display shown above, the following is now true:





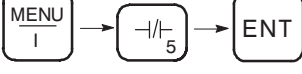
- Output O/1 is forced On; however, the instruction state box is not filled because the instruction is not logically true.
- The controller's output LED is on.
- The controller's Forced I/O LED is on continuously.

Guide to Forcing External Output Circuits

The following occurs in the Run mode:

- The program is scanned and control logic is applied.
- Instruction state boxes are filled for true bit instructions.
- Controller output LEDs go on and are maintained for all forced external output circuits.

The table below shows the keys and key sequences involved with setting and clearing forces. A confirmation screen appears following each of these.

| Key or Key Sequence | Operation |
|--|--|
|  | If the controller is in RRUN, the bit is forced Off and the output circuit remains forced until the force is removed. The data file bit is unaffected. |
|  | If the controller is in RRUN, the bit is forced On and the output circuit remains forced until the force is removed. The data file bit is unaffected. |
|  or  | Affects the cursored external output circuit. It removes the set force. Other forces are unaffected. |
|  | Affects all forced external input bit addresses and external output circuits. It removes set forces from all external input bit addresses and output circuits. |

The displays below show an example of setting a force while in Run mode.

| | | | |
|---------|-----|--------------------------|---|
| R 0 0 1 | () | <input type="checkbox"/> | 0 |
| O / 1 | | | |

Initial conditions.
Bit is Off.
No forces exist.

| | | | |
|---------|-----|-------------------------------------|---|
| R 0 0 1 | () | <input checked="" type="checkbox"/> | 1 |
| O / 1 | | | |

Bit is On.
No forces exist.

| | | | |
|---------|-----|--------------------------|-------|
| R 0 0 1 | () | <input type="checkbox"/> | F O N |
| O / 1 | | | 0 |

Bit is Off. (Set forces do not affect output data file bits.)
Force is On.

| | | | |
|---------|-----|--------------------------|-------|
| R 0 0 1 | () | <input type="checkbox"/> | F O F |
| O / 1 | | | 0 |

Bit is Off.
Force is Off.

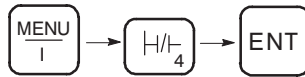
Common Procedures

Several of the menu items may be required after the control program has been running. This chapter gives you details about these procedures and other commonly performed tasks. The common procedures included here are:

- using a memory module
- clearing a program from the micro controller
- changing the micro controller's baud rate
- changing the micro controller's communication defaults

Using a Memory Module

You can use a memory module to store and retrieve programs. To use the memory module, access the menu and choose the option MEM MODULE.



The options available for use with the memory module appear. You can scroll through the options using the down arrow key. These options are:

- load program (from the memory module to the controller)
- store program (from the controller to the memory module)
- clear program (in the memory module)

Each of these options is described in this section. For information on installing and removing the memory module, see page 4-3.



ATTENTION: Always remove power from the HHP before inserting or removing the memory module. This guards against possible damage to the module, as well as undesired controller faults.

Retrieving a Program from a Memory Module

You can retrieve a program from a memory module using the `LOAD PROGRAM` option. Follow the steps below.

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Go to the menu and choose the option `MEM MODULE`.
3. Select `LOAD PROGRAM`.

```
LOAD : → PROGRAM 1
        PROGRAM 2
```

A sub-menu appears listing the names of the programs contained on the memory module. If there are no programs, dashed lines appear in place of program names. In such a case, since there are no programs to load, you must press `ESC` to exit the sub-menu.

4. Arrow down to the program you want to load and select it.



```
LOAD : → PROGRAM 3
LOADING . . .
```

Various screens appear to indicate the program is loading. Once the program is loaded to the controller, the MicroLogix 1000 HHP returns you to the home screen. (If the program is password protected, you are first prompted to enter the password.)

Important: If a program already exists in the controller with the same name as the one you are about to load, a confirmation screen appears asking if you want to overwrite the existing program.

If you do not want to write over the existing program, press `ESC` to exit the sub-menu.

If you want to overwrite the existing program, press `ENT`. Various screens appear to indicate the program is loading. Once the program is loaded to the controller, the MicroLogix 1000 HHP returns you to the home screen. (If the program is password protected, you are first prompted to enter the password.)

Storing a Program to a Memory Module

You can store the program you currently have loaded in the controller to a memory module. Follow the steps below.

1. Put the controller in RPRG mode (if it is not already in that mode).
2. Go to the menu and choose the option MEM MODULE.
3. Select STORE PROGRAM.

```
STORE : → PROGRAM1  
MEMMOD FREE : 008K
```

The name of the program and the amount of memory remaining in the module is shown.

4. Begin the storing process.

```
ENT
```

```
STORE : → PROGRAM1  
CHECKING . . .
```

Important: If edits exist in the program, the program is verified for errors (saved in the controller) before it is stored in the memory module. Programs with errors cannot be saved in the controller or stored in the memory module.

Various screens appear to indicate the program is being stored to the memory module. Once the program is stored, you are returned to the memory module sub-menu.

Important: If an OVERWRITE PROG? confirmation screen or a MEMMOD FULL message appears, see the appropriate section below.

If an OVERWRITE PROG? Confirmation Screen Appears

If a program already exists in the memory module with the same name as the one you are about to store, a confirmation screen appears asking if you want to overwrite the existing program. (The memory module acts like a floppy disk.)

If you want to write over the existing program, press **ENT**. Various screens appear to indicate the program is being stored to the memory module. Once the program is stored, you are returned to the memory module sub-menu.

If you do not want to write over the existing program, press **ESC** to exit the sub-menu. Then if you still want to store the program that is currently loaded in the controller to a memory module, do *one* of the following:

- Go to the program configuration menu and rename the current program (see page 18–2). Then store the program to the memory module under its new name.
- Replace the memory module that is currently in the MicroLogix 1000 HHP with one that does not have a program with this name. Then store the program to that memory module.



ATTENTION: Always remove power from the HHP before inserting or removing the memory module. This guards against possible damage to the module, as well as undesired controller faults.

If a **MEM MOD FULL** Message Appears

If the module does not have sufficient memory to store the program, the message **MEM MOD FULL** appears. Press any key to remove the message. You are then returned to the memory module options screen.

If you still want to store the program that is currently loaded in the controller to a memory module, do *one* of the following:

- Clear a program(s) from the memory module to make room for the program that is currently loaded. (See the next section.) Then try storing the program to the memory module again.
- Replace the memory module that is currently in the MicroLogix 1000 HHP with one that has more memory available. Then store the program to that memory module.



ATTENTION: Always remove power from the HHP before inserting or removing the memory module. This guards against possible damage to the module, as well as undesired controller faults.

Clearing a Program from a Memory Module

You can clear a program currently stored on a memory module by following the steps below:

1. Go to the menu and choose the option `MEM MODULE`.
2. Select `CLEAR PROGRAM`.

```
CLEAR : → PROGRAM 1
          PROGRAM 2
```

A sub-menu appears listing the names of the programs contained on the memory module. If there are no programs, dashed lines appear in place of program names. In such a case, since there are no programs to clear, you must press `ESC` to exit the sub-menu.

3. Arrow down to the program you want to clear and select it.



```
CLEAR PROGRAM ?
PROGRAM 3 : 002K
```

A confirmation screen appears that shows the name of the program and its size. If you do not want to clear this program, press `ESC` to exit the sub-menu.

4. Clear the program in the memory module.



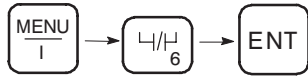
```
CLEAR : → PROGRAM 3
CLEARING . . .
```

Once the program is cleared from the module, you are returned to the memory module sub-menu.

Clearing a Program from the Micro Controller

If you want to create a new program, you need to first clear the current program from the controller. Follow the steps below.

1. Enter the menu and choose the option `CLEAR PROG.`



```
CLEAR PROGRAM?  
YES [ENT] NO [ESC]
```

2. Clear the program from the controller.



```
CLEAR PROGRAM?  
CLEARING . . . . .
```

Once the program is cleared, the menu options are displayed. You can return to the previous functional areas by pressing `ESC`.

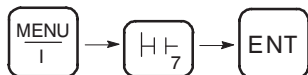
Important: All program configuration settings are returned to their defaults. See page 18–1 for information of changing the default settings.

Changing the Micro Controller's Baud Rate

Follow the steps below to change the controller's baud rate. (The default is 9600 baud.)

Important: These steps apply only to programs configured for Series A and B MicroLogix 1000 discrete controllers. If you have configured your program for Series C or later discrete controllers or for MicroLogix analog controllers, see page 19–7 for directions on changing the baud rate.

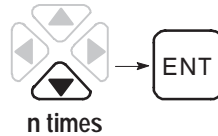
1. Enter the menu and choose the option `BAUD RATE.`



```
→ 9 6 0 0  
   2 4 0 0
```

The arrow points to the controller's current baud rate.

2. Arrow down to the desired baud rate and select it.



The MicroLogix 1000 HHP resets itself to the new baud rate and runs through the power-up sequence.

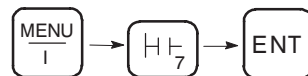
Changing the Micro Controller's Communication Defaults

If you configure your program to be compatible with Series C or later MicroLogix 1000 discrete controllers or MicroLogix 1000 analog controllers, option number 7 in the menu functional area is COMMS (instead of BAUD RATE). This menu option allows you to change any or all of the following:

- the controller's active communication protocol
- the default baud rates for each protocol
- the default node addresses for each protocol

The steps below walk you through the five sub-menus of the COMMS option. Press **ESC** at any time to exit the sub-menus.

1. Enter the menu and choose the option COMMS. The first screen that appears is the Main Protocol sub-menu, as shown below.



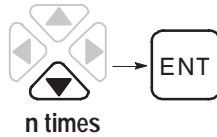
Use this sub-menu to specify which protocol the controller should attempt to establish communication with first. Although the MicroLogix 1000 controller will *automatically* find which protocol is active (DF1 full-duplex or DH-485) and establish communication accordingly, you can shorten the connection time by choosing the appropriate main protocol.

2. The default setting for the main protocol is DF1.
 - To *accept* the default setting (DF1) , press ENT.
 - To *change* the default setting, arrow right to select DH-485 and then press ENT.

The arrow points to the controller’s current DF1 baud rate.

| | | |
|---|---------|------|
| → | 9 6 0 0 | DF 1 |
| | 2 4 0 0 | DF 1 |

3. Arrow down to the desired DF1 baud rate and select it.



4. The DF1 node address sub-menu appears next.

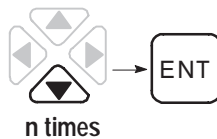
| | | |
|--------|------|-----------|
| DF 1 | NODE | 0 - 2 5 4 |
| ADDR : | 1 | ■ |

- To *accept* the default setting (1), press ENT.
- To *change* the default setting, delete the 1 by pressing DEL, type in the new number, and then press ENT.

The arrow points to the controller’s current DH-485 baud rate.

| | | |
|---|-----------|----------|
| → | 1 9 2 0 0 | DH 4 8 5 |
| | 9 6 0 0 | DH 4 8 5 |

5. Arrow down to the desired DH-485 baud rate and select it.



6. The DH-485 node address sub-menu appears next.

```
DH485 NODE 1-31
ADDR: 1 ■
```

- To *accept* the default setting (1), press ENT.
- To *change* the default setting, delete the 1 by pressing DEL, type in the new number, and then press ENT.

```
APPLY COMMS
YES [ENT] NO [ESC]
```

7. Accept the communication settings.

ENT

```
APPLYING COMMS
RESETTING UNIT
```

The MicroLogix 1000 HHP resets itself to the new communication settings and runs through the power-up sequence.

Note: You cannot use the MicroLogix 1000 HHP to select or configure the DF1 half-duplex slave protocol for Series D discrete or Series A analog MicroLogix 1000 controllers.

Troubleshooting Your System

This chapter describes how to troubleshoot your controller. Topics include:

- understanding the controller LED status
- identifying HHP errors
- using the trace feature
- controller error recovery model
- identifying controller faults
- recovering your work
- calling Allen-Bradley for assistance

Understanding the Controller LED Status

Between the time you apply power to the controller and the time it has to establish communication with a connected programming device, the only form of communication between you and the controller is through the LEDs.

When Operating Normally

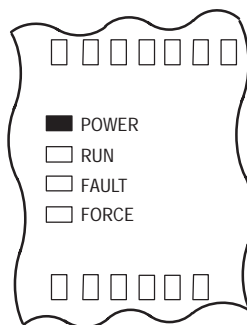
When power is applied, only the power LED turns on and remains on, as shown on the left in the figure below. This is part of the normal power up sequence.

When the controller is placed in RRUN mode, the run LED also turns on and remains on, as shown on the right in the figure below. If a force exists, the force LED is on as well.

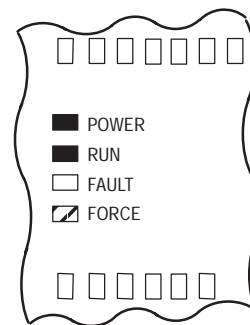
Refer to the following key to determine the status of the LED indicators:

- Indicates the LED is OFF.
- Indicates the LED is ON.
- Indicates the LED is FLASHING.
- Status of LED does not matter.

When powered up:



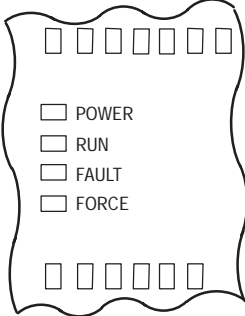
When placed in RRUN:



When an Error Exists

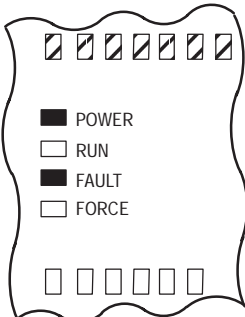
If an error exists within the controller, the controller LEDs operate as described in the following tables.

If the LEDs indicate:



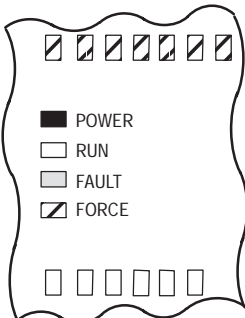
| The Following Error Exists | Probable Cause | Recommended Action |
|--------------------------------------|-------------------------|---|
| No input power or power supply error | No Line Power | Verify proper line voltage and connections to the controller. |
| | Power Supply Overloaded | This problem can occur intermittently if power supply is overloaded when output loading and temperature varies. |

If the LEDs indicate:



| The Following Error Exists | Probable Cause | Recommended Action |
|----------------------------|-------------------------|---|
| Hardware fault | Controller Memory Error | Cycle power. Contact your local Allen-Bradley representative if the error persists. |
| | Loose Wiring | Verify connections to the controller. |

If the LEDs indicate:



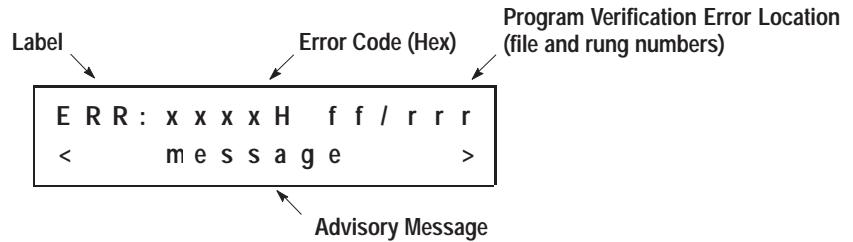
| The Following Error Exists | Probable Cause | Recommended Action |
|----------------------------|--|---|
| Application fault | Hardware/Software Major Fault Detected | <ol style="list-style-type: none"> 1. Monitor Status File Word S6 for major error code. 2. Remove hardware/software condition causing fault. 3. Use the hot key sequence FAULT-DEL to clear the fault. 4. Attempt to enter RRUN, RCSN, or RSSN mode. If unsuccessful, repeat recommended action steps above or contact your local Allen-Bradley distributor. |

Refer to the following key to determine the status of the LED indicators:

- Indicates the LED is OFF.
- Indicates the LED is ON.
- Indicates the LED is FLASHING.
- Status of LED does not matter.

Identifying HHP Errors

When an error occurs while accepting edits, the MicroLogix 1000 HHP displays an error message screen as shown here:



Use the tables in this section to find the error code and the associated recommended action(s) you should take to clear the error. The tables are grouped by error type as described in the table below.

| If the error code begins with a | then the error type is |
|---------------------------------|------------------------|
| 0 | Hardware |
| 1, 5, 6, or F | Communication |
| 2 | Miscellaneous |
| 3 | Program Verification |

Hardware Error Messages

| Error Code (Hex) | Advisory Message | Description | Recommended Action |
|------------------|------------------|--|--|
| 0001 | HARDWARE FAILURE | The HHP encountered a ROM, RAM, or keypad failure. | Disconnect the HHP from the micro controller, then reconnect it. If the error persists, record the error code and contact your local Allen-Bradley representative. |
| 0002 | | | |
| 0003 | | | |
| 0004 | NO CONTROLLER | The HHP cannot establish a communication link to the micro controller. | Disconnect the HHP from the micro controller, then reconnect it. If the error persists, record the error code and contact your local Allen-Bradley representative. |
| 0006 | MEM MOD MISSING | The HHP cannot establish a connection to the memory module. | Disconnect the HHP from the micro controller. Then either reconnect it, or replace the memory module in the HHP with a new one. |

Communication Error Messages

| Error Code (Hex) | Advisory Message | Description | Recommended Action |
|--|---------------------|--|--|
| 1000 | ILLEGAL COMMAND | A communication error has occurred between the HHP and the micro controller. | Disconnect the HHP from the micro controller, then reconnect it. If the error persists, record the error code and contact your local Allen-Bradley representative. |
| 5000 | ILLEGAL ADDRESS | During download the HHP has written data beyond a file boundary in the micro controller. | Disconnect the HHP from the micro controller, then reconnect it. If the error persists, record the error code and contact your local Allen-Bradley representative. |
| 6000 | NOT ALLOWED | A communication error has occurred between the HHP and the micro controller. | Disconnect the HHP from the micro controller, then reconnect it. If the error persists, record the error code and contact your local Allen-Bradley representative. |
| F00B | ACCESS DENIED | The micro controller must be in program mode. | Use the HHP to change the controller's mode to RPRG and try the operation again. |
| F00C | NOT AVAILABLE | The micro controller must be in program mode. | Use the HHP to change the controller's mode to RPRG and try the operation again. |
| F01A | FILE OPEN | A communication error occurred between the HHP and the micro controller. | Disconnect the HHP from the micro controller, then reconnect it. If the error persists, record the error code and contact your local Allen-Bradley representative. |
| F01B | PROGRAM OWNED | A communication error occurred between the HHP and the micro controller. | Disconnect the HHP from the micro controller, then reconnect it. If the error persists, record the error code and contact your local Allen-Bradley representative. |
| all other 1xxx, 5xxx, 6xxx, or Fxxx codes | CONTROLLER ERROR | A communication error occurred between the HHP and the micro controller. | Disconnect the HHP from the micro controller, then reconnect it. If the error persists, record the error code and contact your local Allen-Bradley representative. |

Miscellaneous Error Messages

| Error Code (Hex) | Advisory Message | Description | Recommended Action |
|------------------|------------------|--|--|
| 2000 | NO RESPONSE | A communication error occurred between the HHP and the micro controller. | Disconnect the HHP from the micro controller, then reconnect it. If the error persists, record the error code and contact your local Allen-Bradley representative. |
| 2001 | INVALID DEVICE | The device that the HHP is attached to is telling the HHP that it is not a micro controller. | Make sure your HHP is connected to a micro controller. If the error persists, record the error code and contact your local Allen-Bradley representative. |

Program Verification Error Messages

| Error Code (Hex) | Advisory Message | Description | Recommended Action |
|------------------|------------------|--|---|
| 3000 | MISSING SOR | The rung does not begin with a start of rung instruction. | Add a start of rung instruction to the beginning of the rung using the NEW RUNG key. |
| 3001 | INVALID INPUT | The first input instruction on the rung is not a load instruction (e.g., LD, LDI, LD EQU). | Change the input instruction immediately after the Start of Rung instruction to a load instruction. |
| 3002 | MISSING OUTPUT | The rung does not end with an output instruction. | Add the appropriate output instruction to the end of the rung. |
| 3003 | INVALID RUNG | An instruction or instruction block needs an ANB or ORB to properly join it with the previous input logic on the rung. | Check that all instructions or instruction blocks have been properly joined to the previous input logic on the rung with an ANB or ORB. |
| 3004 | INVALID RUNG | An MPP is missing from the rung. | Add an MPP at the proper location. |
| 3005 | INVALID ORB | An ORB instruction is illegally positioned on the rung. | Move the ORB to the correct location on the rung or remove it if the application does not require it. |
| 3006 | INVALID ANB | An ANB instruction is illegally positioned on the rung. | Move the ANB to the correct location on the rung or remove it if the application does not require it. |
| 3007 | INVALID ORB, OR | The position of an OR or ORB instruction after an MPS, MRD, or MPP instruction has created an illegal rung. | Review and rewrite the instruction list logic. Two rungs may be required to implement the desired functionality. |

| Error Code (Hex) | Advisory Message | Description | Recommended Action |
|------------------|------------------|--|--|
| 3008 | INVALID MRD, MPP | MRD and/or MPP instructions are not preceded by an output instruction. | Ensure that each MRD and MPP instruction is preceded by an output instruction or remove them if the application does not require them. |
| 3009 | INVALID MRD, MPS | An MRD instruction is used illegally. | Ensure that the MRD instruction is preceded by an MPS instruction or remove the MRD if the application does not require it. |
| 300A | INVALID MPP, MPS | An MPP instruction is used illegally. | Ensure that the MPP instruction is preceded by an MPS instruction or remove the MPP if the application does not require it. |
| 300B | INVALID LDT | The LDT instruction is used illegally. | Ensure that the LDT is followed by an OR or ORB instruction on the rung or remove the LDT if the application does not require it. |
| 300C | INVALID HSC | An instruction is using the data table address C0 when an HSC instruction is present in the program. | Change the instruction's address to something other than C0 or remove the HSC instruction if the application does not require it. |
| 300D | INVALID SBR, INT | The SBR or INT instruction is used illegally. | Move the SBR or INT instruction so it is the first instruction on the first rung of the program file or remove the instruction if the application does not require it. |
| 300E | INVALID LBL | The LBL instruction is used illegally. | Move the LBL instruction so it is the first instruction on the rung or remove the instruction if the application does not require it. |
| 300F | INVALID FOR FILE | An SBR, INT, or RET instruction exists in program file 2. | Move the instruction to the correct file or remove it if the application does not require it. |
| 3010 | INVALID OSR | An OSR instruction is illegally positioned on the rung. | Move the instruction to a correct position on the rung or remove the instruction if the application does not require it. |
| 3011 | INVALID COMPARE | A comparison instruction has a constant value as the first operand. | Change the first operand of the instruction so it is not a constant value or remove the instruction if the application does not require it. |
| 3012 | INVALID ADDRESS | An instruction is referencing an address outside of the data table space. | Ensure that the operands for each instruction are within the micro controller's data file space. |

| Error Code (Hex) | Advisory Message | Description | Recommended Action |
|-------------------------|-------------------------|--|--|
| 3013 | INVALID ADDRESS | An instruction is referencing a status file address outside of the data table space. | Ensure that the status file operands for each instruction are within the micro controller's data file space. |
| 3014 | BRANCH NEST ERR | The program's structure exceeds the allowable number of nested branches (4). | Ensure that the program contains no more than 4 nested MPS instructions at a time, as each MPS counts toward a nested branch. |
| 3015 | BRANCH LEVEL ERR | The program's structure exceeds the allowable number of levels per nested branch (75). | Ensure that for each MPS/MPP pair, there are no more than 73 MRDs within them. |
| 3016 | TOO MANY INST | A rung contains more than the allowable number of instructions (128). | Redesign the program file so there are no more than 128 instructions on each rung. |
| 3017 | TOO MANY HSC'S | The program contains more than the allowable number of HSC instructions (1). | Remove the extra HSC instructions so only one exists in the program or divide the program so that each HSC instruction is in a separate program. |
| 3018 | UNMATCHED MCR'S | The program is missing either a starting or ending MCR instruction. | Ensure that the program has a starting and an ending MCR instruction or remove the MCR instruction if the application does not require it. |
| 3019 | INVALID MCR | The program's ending MCR is illegally placed on a rung that contains other instructions. | Rearrange the logic so the MCR instruction is the only instruction on a rung. |
| 301A | INVALID FILE | The file number specified by a JSR instruction is outside the allowable range of program files (3–15). | Specify a program file between 3 and 15. |
| 301B | INVALID RES | A RES instruction resets a timer address previously used by a TOF instructions. | Change the RES or TOF address or remove one if the application does not require it. |
| 301C | INVALID TOF | A TOF instruction specifies a timer address previously reset by a RES instruction. | Change the TOF or RES address or remove one if the application does not require it. |
| 301D | INVALID LBL | The same label number is specified by more than one LBL instruction. | Give the LBL instruction a unique label number or remove the instruction if the application does not require it. |

| Error Code (Hex) | Advisory Message | Description | Recommended Action |
|------------------|------------------|--|---|
| 301E | INVALID JMP | A JMP instruction specifies a label number for which no corresponding LBL instruction exists. | Ensure that a valid LBL instruction and label number exist in the program or remove the instruction if the application does not require it. |
| 301F | INVALID ADDRESS | A file instruction exceeds the allowable data file size. | Ensure that the address of the file plus the length does not go beyond the data file for the address specified or remove the instruction if the application does not require it. |
| 3023 | INVALID W/O HSC | The program does not contain an HSC instruction, yet has an OUT instruction with the address of C0/UA. | Either change the address of the OUT instruction, add an HSC instruction or remove the OUT instruction if the application does not require it. |
| 3024 | INVALID ADDRESS | A bit specific to the HSC instruction is used illegally in the program. | Make sure that the bits specific to the HSC operation are from data file address C0. Also, if any other instructions have bit references from data file address C0, change those references to a different data file. |
| 3Fxx | INTERNAL ERROR | An error has occurred within the HHP. | Record the error code and contact your local Allen-Bradley representative. |

Using the Trace Feature

The trace feature helps you locate faulty inputs that prevent outputs from turning on or off as they should. The trace feature will find the following:

- output coils (OUT, RST, and SET)
- timer, counter, and control instructions (BSL, BSR, CTD, CTU, FFL, FFU, HSC, HSD, HSE, HSL, LFL, LFU, RAC, RES, RTO, SQC, SQL, SQO, TOF, TON)

For output coils and timer, counter, and control instructions, the trace begins from the current location in the program and continues to the end of the program. For all other instructions, the trace begins from the top of the first program file and traces down through all of the files.

If the address you are tracing is found, you can invoke the trace feature once again to search for other instances of the same address. For all instructions, the trace begins from the current location in the program and continues to the end of the program. When the last program file is reached, the trace feature does not wrap around to the first program file.

If at any time the trace feature cannot find the address, the message NOT FOUND is displayed.

You can execute trace for either an address that you enter or an address that is displayed.

Tracing an Address You Enter

To enter an address and trace it, press the key sequence shown here:



This type of tracing method can be invoked from any of the four functional areas:

- home screen
- program monitor
- data monitor
- multi-point

Tracing an Address That is Displayed

You can trace the address that is currently displayed on the MicroLogix 1000 HHP by pressing the key shown here:



You can invoke trace using this method from the following functional areas:

- program monitor
- data monitor
- multi-point

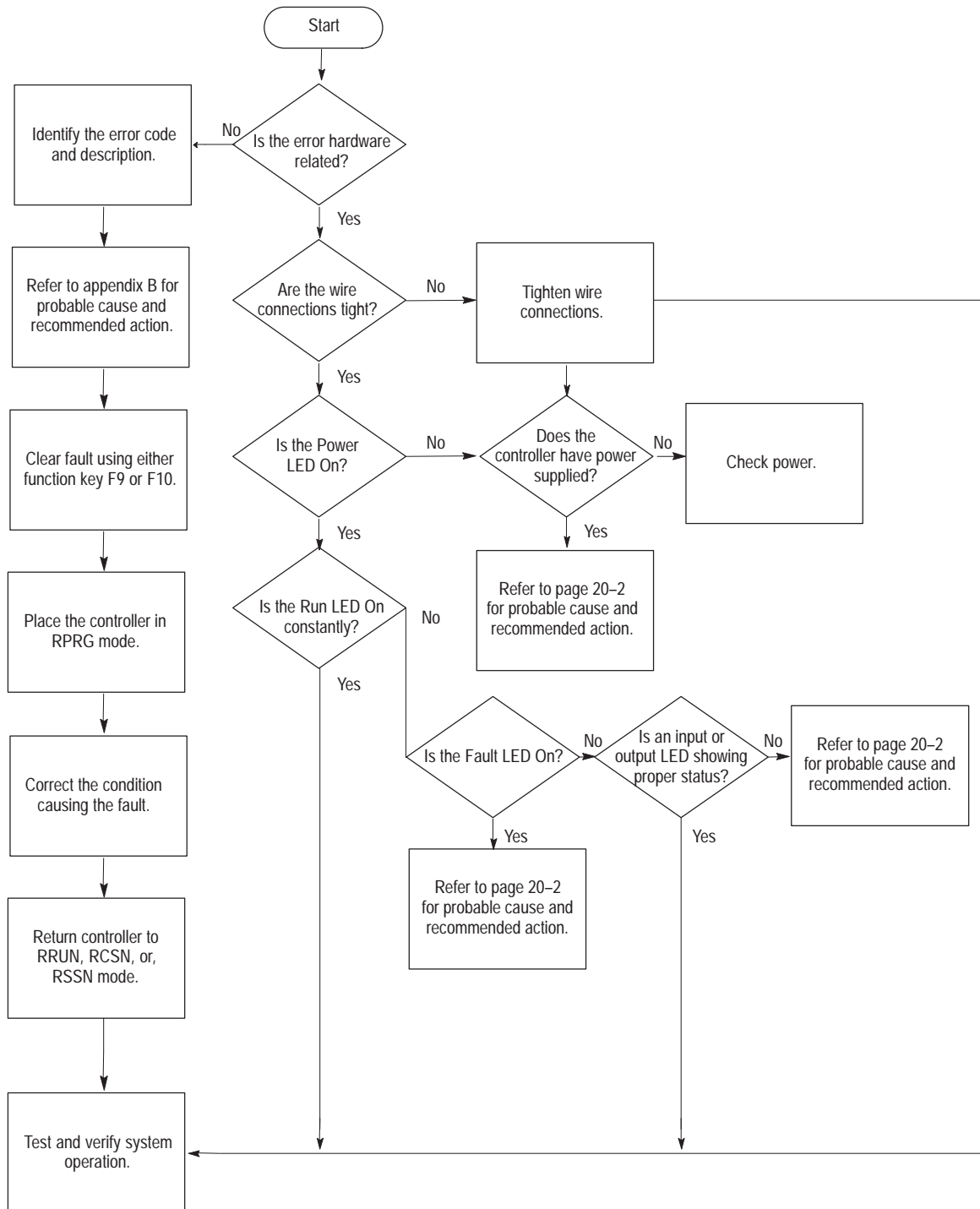
Tracing Bit Addresses Versus Word Addresses

The following table outlines whether the trace feature finds a bit address or a word address when invoked from any of the functional areas.

| If the address entered or displayed is a(n): | Then the trace finds: |
|--|--|
| output, bit, integer, or status file address | only bit instructions referencing that bit. To invoke the trace feature, the current radix must be <i>binary</i> . |
| timer, counter, or control address | any bit instructions or parameters of word instructions referencing that Timer, Counter or Control element |

Controller Error Recovery Model

Use the following error recovery model to help you diagnose software and hardware problems in the micro controller. The model provides common questions you might ask to help troubleshoot your system. Refer to the recommended pages within the model and to S6 of the status file on page B-8 for further help.



Identifying Controller Faults

While a program is executing, a fault may occur within the operating system or your program. When a fault occurs, you have various options to determine what the fault is and how to correct it. This section describes how to clear faults and provides a list of possible advisory messages with recommended corrective actions.

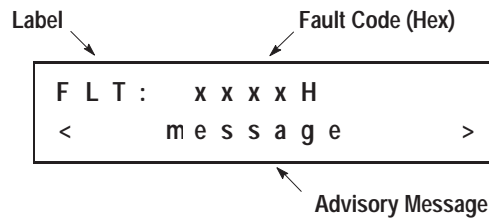
Clearing Controller Faults

There are three ways you can clear a fault:

- manually, by viewing the fault display
- automatically, using the Fault Override bit (S1/8) or the Run Always bit (S1/12) and cycling power
- implementing the user-fault routine and clearing bit S1/13 within it

Manually Clearing Faults

You can manually view and clear a fault by accessing the fault display.



Follow these steps to manually clear a fault:

1. From any of the functional areas, access the fault display.



```
FLT : 0 0 2 0 H
MINOR ERR AT END
```

If you press this key and a fault does not exist, the message NO MAJOR FAULT is displayed. Press any key to return to the previous screen.

2. Clear the fault.



```
F L T :   0 0 2 0 H
C L E A R I N G . . . . .
```

After the fault is cleared, the HHP returns to the screen that was displayed prior to accessing the fault display.

Automatically Clearing Faults

You can automatically clear a fault when cycling power to the controller by setting either one or both of the following status bits in the status file:

- Fault Override at Powerup bit (S1/8)
- Run Always bit (S1/12)



ATTENTION: Clearing a fault using the Run Always bit (S1/12) causes the controller to immediately enter the RRUN mode. Make sure you fully understand the use of this bit before incorporating it into your program. Refer to page B-4 for more information. Also refer to chapter 5 for information pertaining to retentive data.

Refer to appendix B for more information on status bits.

Important: You can declare your own application-specific major fault by writing your own unique value to S6 and then setting bit S1/13 to prevent reusing system defined codes. The recommended values for user defined faults is FF00 to FF0F.

Using the Fault Routine

The occurrence of recoverable or non-recoverable user faults causes file 3 to be executed. If the fault is recoverable, the subroutine can be used to correct the problem and clear the fault bit S1/13. The controller then continues in the RRUN mode.

The subroutine does not execute for non-user faults. The user-fault routine is discussed in chapter 7.

Controller Fault Messages

This section contains controller fault messages that the MicroLogix 1000 HHP may display during operation. Refer to page B–9 for a listing of recoverable and non-recoverable faults.

| Fault Code (Hex) | Advisory Message | Description | Recommended Action |
|------------------|------------------|---|---|
| 0001 | DEFAULT LOADED | The default program is loaded to the controller memory. This occurs: <ul style="list-style-type: none"> on power up if the power down occurred in the middle of a program save or a memory module load if the user program is corrupt at power up | <ol style="list-style-type: none"> Re-save or re-load the program and enter the RRUN, RCSN, or RSSN mode. Contact your local Allen-Bradley representative if the error persists. |
| 0002 | UNEXPECTED RESET | The controller was unexpectedly reset due to a noisy environment or internal hardware failure. If the user program previously saved in the controller or loaded from the memory module is valid, the data previously saved in the controller or loaded from the memory module with the program is used. The Retentive Data Lost Bit (S5/8) is set. If the user program is invalid, the default program is loaded. | <ol style="list-style-type: none"> Refer to proper grounding guidelines in chapter 2. Contact your local Allen-Bradley representative if the error persists. |
| 0003 | PROG CORRUPTED | While power cycling to your controller, a noise problem may have occurred. The user program is corrupt and the default program is loaded. | <ol style="list-style-type: none"> Try cycling power again. Your program may be valid, but retentive data is lost. Re-save or re-load your program. Contact your local Allen-Bradley representative if the error persists. |
| 0004 | PROGRAM CHANGED | While the controller was in the RUN mode or any test mode, the ROM or RAM became corrupt. If the user program is valid, the program and initial data previously saved in the controller or loaded from the memory module is used and the Retentive Data Lost Bit (S5/8) is set. If the user program is invalid, error 0003 occurs. | <ol style="list-style-type: none"> Cycle power on your unit. Re-save or re-load your program and re-initialize any necessary data. Start up your system. Contact your local Allen-Bradley representative if the error persists. |
| 0005 | RETENT DATA LOST | The data files (input, output, timer, counter, integer, binary, control, and status) are corrupt. | <ol style="list-style-type: none"> Cycle power on your unit. Re-save or re-load your program and re-initialize any necessary data. Start up your system. Contact your local Allen-Bradley representative if the error persists. |

| Fault Code (Hex) | Advisory Message | Description | Recommended Action |
|------------------|------------------|--|---|
| 0008 | INTERNAL ERROR | The controller software has detected an invalid condition within the hardware or software after completing power up processing (after the first 2 seconds of operation). | <ol style="list-style-type: none"> 1. Cycle power on your unit. 2. Re-save or re-load your program and re-initialize any necessary data. 3. Start up your system. 4. Contact your local Allen-Bradley representative if the error persists. |
| 0009 | INTERNAL ERROR | The controller software has detected an invalid condition within the hardware during power up processing (within the first 2 seconds of operation). | <ol style="list-style-type: none"> 1. Cycle power on your unit. 2. Re-save or re-load your program and re-initialize any necessary data. 3. Start up your system. 4. Contact your local Allen-Bradley representative if the error persists. |
| 0010 | WRONG PROC REV | The program in the controller is not configured for a micro controller. | If you want to use a micro controller with the program, reconfigure your controller using programming software, or clear the program in the controller with the HHP. |
| 0016 | START AFTER P.F. | The system has powered up in the RRUN mode. Bit S1/13 is set and the user-fault routine is run before beginning the first scan of the program. | Either reset bit S1/9 if this is consistent with your application requirements, and change the mode back to RRUN, or clear S1/13, the major fault bit. |
| 0018 | INCOMPAT PROGRAM | An incompatible program is in the controller. Either the program does not have the correct number of files or it does not have the correct size data files. The default program is loaded. | If you want to use a micro controller with the program, reconfigure your controller using programming software, or clear the program in the controller with the HHP. |
| 0020 | MINOR ERR AT END | A minor fault bit (bits 0–7) in S5 was set at the end of scan. | Correct the condition that caused the error, then clear the fault using the FAULT - DEL keys and enter the RRUN, RCSN, or RSSN mode. |
| 0022 | WATCHDOG TIMEOUT | The program scan time exceeded the watchdog timeout value. | Verify if the program is caught in a loop and correct the problem, or increase the watchdog timeout value using the program configuration menu selection. |
| 0024 | INVALID STI TIME | An invalid STI interval exists (not between 0 and 255). | Set the STI interval between the values of 0 and 255 using the program configuration menu selection. |
| 0025 | TOO MANY JSR'S | There are more than 3 subroutines nested in the STI subroutine (file 5). | Correct the user program to meet the requirements and restrictions for the JSR instruction, then re-enter RRUN, RCSN, or RSSN mode. |

| Fault Code (Hex) | Advisory Message | Description | Recommended Action |
|------------------|------------------|--|--|
| 0027 | TOO MANY JSR'S | There are more than three subroutines nested in the fault routine (file 3). | Correct the user program to meet the requirements and restrictions for the JSR instruction, then re-enter RRUN, RCSN, or RSSN mode. |
| 002A | INDEX TOO LARGE | The program is referencing through indexed addressing an element beyond a file boundary. | Correct the user program to not index beyond file boundaries. |
| 002B | TOO MANY JSR'S | There are more than three subroutines nested in the high-speed counter routine (file 4). | Correct the user program to meet the requirements and restrictions for the JSR instruction, then re-enter RRUN, RCSN, or RSSN mode. |
| 0030 | SUB NEST DEPTH | There are more than eight subroutines nested in the main program file (file 2). | Correct the user program to meet the requirements and restrictions for the main program file, then re-enter RRUN, RCSN, or RSSN mode. |
| 0031 | UNSUPPORTED INST | The program contains an instruction(s) that is not supported by the micro controller. For example MSG, SVC, or PID. | Modify the program so that all instructions are supported by the controller, then reload the program and enter the RRUN, RCSN, or RSSN mode. |
| 0032 | INVALID SQx LEN | A sequencer instruction length/position parameter points past the end of a data file. | Correct the program to ensure that the length and position parameters do not point past the data file. Then re-enter RRUN, RCSN, or RSSN mode. |
| 0033 | INVALID BSx LEN | The length parameter of a BSL, BSR, FFL, FFU, LFL, or LFU instruction points past the end of a data file. | Correct the program to ensure that the length parameter does not point past the data file. Reload the program and enter the RRUN, RCSN, or RSSN mode. |
| 0034 | INVALID TIMER | A negative value was loaded to a timer preset or accumulator. | If the program is moving values to the accumulator or preset word of a timer, make certain these values are not negative. Correct the program and re-enter RRUN, RCSN, or RSSN mode. |
| 0035 | INVALID FOR FILE | The program contains a Temporary End (TND) instruction in file 3, 4, or 5 when it is being used as an interrupt subroutine. | Correct the program and re-enter RRUN, RCSN, or RSSN mode. |
| 0037 | INVALID HSC PRE | Either a zero (0) or a negative high preset was loaded to counter (C0) when the HSC was an Up counter or the high preset was lower than or equal to the low preset when the HSC was a Bidirectional counter. | <ol style="list-style-type: none"> 1. Check to make sure the presets are valid. 2. Correct the program and re-enter RRUN, RCSN, or RSSN mode. |

| Fault Code (Hex) | Advisory Message | Description | Recommended Action |
|-------------------|------------------|---|--|
| 0038 | RET IN FILE 2 | A RET instruction is in the main program file (file 2). | Remove the RET instruction and re-enter RRUN, RCSN, or RSSN mode. |
| 0040 | OUTPUT VERIFY WR | When outputs were written and read back by the controller, the read failed. This may have been caused by noise. | <ol style="list-style-type: none"> 1. Refer to proper grounding guidelines in chapter 2. 2. Start up your system. 3. Contact your local Allen-Bradley representative if the error persists. |
| 0041 ^① | EXTRA OUTPUT SET | An extra output bit was set when the Extend I/O Configuration bit in the program configuration menu was reset. For 16-point controllers this includes bits 6–15. For 32-point controllers this includes bits 12–15. | <ol style="list-style-type: none"> 1. Set the Extend I/O bit or change your application to <i>prevent</i> these bits from being turned on. 2. Correct the program and re-enter the RRUN, RCSN, or RSSN mode. |

^① Valid for Series A – C discrete only.

Recovering Your Work

If the MicroLogix 1000 HHP is disconnected or power failure occurs, the HHP retains any edits you made to the program before power was removed. To recover your program with edits, reconnect the HHP to the controller you were using when power was lost.

Important: If you connect the MicroLogix 1000 HHP to a controller other than the one you were using while performing program edits, you will lose your edits.

At the end of the power-up sequence the HHP displays the screen shown below.

Program name

M I C R O
E D I T S E X I S T

This screen indicates that edits have been made to the program but have not yet been saved.

Important: If power is removed for more than three days, the retained program edits may be lost.

Calling Allen-Bradley for Assistance

If you need to contact Allen-Bradley or your local distributor for assistance, it is helpful to obtain the following (prior to calling):

- controller type, series letter, firmware (FRN) number (see label on side of controller)
- controller LED status
- controller error codes (found in S6 of status file)
- revision of programming device (on power sequence display of the HHP)

Hardware Reference

This appendix lists the MicroLogix 1000 Programmable Controller and MicroLogix 1000 HHP:

- specifications
- dimensions
- accessories and replacement parts

For AIC+ specifications, see the *Advanced Interface Converter User Manual*, Publication 1761-6.4. For DNI specifications, see the *DeviceNet Interface™ User Manual*, Publication 1761-6.5.

Controller Specifications

Controller Types

| Catalog Number | Description |
|----------------|--|
| 1761-L16AWA | 10 pt. ac input, 6 pt. relay output, ac power supply controller |
| 1761-L32AWA | 20 pt. ac input, 12 pt. relay output, ac power supply controller |
| 1761-L20AWA-5A | 12 pt. ac input, 4 pt. analog input, 8 pt. relay output, 1 pt. analog output, ac power supply controller |
| 1761-L10BWA | 6 pt. dc input, 4 pt. relay output, ac power supply controller |
| 1761-L16BWA | 10 pt. dc input, 6 pt. relay output, ac power supply controller |
| 1761-L20BWA-5A | 12 pt. dc input, 4 pt. analog input, 8 pt. relay output, 1 pt. analog output, ac power supply controller |
| 1761-L32BWA | 20 pt. dc input, 12 pt. relay output, ac power supply controller |
| 1761-L10BWB | 6 pt. dc input, 4 pt. relay output, dc power supply controller |
| 1761-L16BWB | 10 pt. dc input, 6 pt. relay output, dc power supply controller |
| 1761-L20BWB-5A | 12 pt. dc input, 4 pt. analog input, 8 pt. relay output, 1 pt. analog output, dc power supply controller |
| 1761-L32BWB | 20 pt. dc input, 12 pt. relay output, dc power supply controller |
| 1761-L16BBB | 10 pt. dc input, 4 pt. FET and 2 pt. relay outputs, dc power supply controller |
| 1761-L32BBB | 20 pt. dc input, 10 pt. FET and 2 pt. relay outputs, dc power supply controller |
| 1761-L32AAA | 20 pt. ac input, 10 pt. triac and 2 pt. relay outputs, ac power supply controller |

General Specifications

| Description: | | Specification: 1761-L | | | | | | | | | | | | |
|--|--|-----------------------|----------|-------|-------------|-------|----------|-------|----------------|----------------|--------------|--------------|----------|-------|
| | | 16AWA | 20AWA-5A | 32AWA | 10BWA | 16BWA | 20BWA-5A | 32BWA | 32AAA | 16BBB | 10BWB | 16BWB | 20BWB-5A | 32BWB |
| Memory Size and Type | 1 K EEPROM (approximately 737 instruction words: 437 data words) | | | | | | | | | | | | | |
| Power Supply Voltage | 85–264V ac, 47-63 Hz | | | | | | | | 20.4–26.4V dc | | | | | |
| Power Supply Usage | 120V ac | 15 VA | 20 VA | 19 VA | 24 VA | 26 VA | 30 VA | 29 VA | 16 VA | Not Applicable | | | | |
| | 240V ac | 21 VA | 27 VA | 25 VA | 32 VA | 33 VA | 38 VA | 36 VA | 22 VA | | | | | |
| | 24V dc | Not Applicable | | | | | | | | 5W | | 10W | 7W | |
| Power Supply Maximum Inrush Current ^① | 30A for 8 ms | | | | | | | | 30A for 4 ms | | 50A for 4 ms | 30A for 4 ms | | |
| 24V dc Sensor Power (V dc at mA) | Not Applicable | | | | 200 mA | | | | Not Applicable | | | | | |
| Max Capacitive Load (User 24V dc) | | | | | 200 μ F | | | | | | | | | |
| Power Cycles | 50,000 minimum | | | | | | | | | | | | | |
| Operating Temp. | Horizontal mounting: 0°C to +55°C (+32°F to +131°F) for horizontal mounting Vertical mounting ^② : 0°C to +45°C (+32°F to +113°F) for discrete; 0°C to +40°C (+32°F to +113°F) for analog | | | | | | | | | | | | | |
| Storage Temp. | –40°C to +85°C (–40°F to +185°F) | | | | | | | | | | | | | |
| Operating Humidity | 5 to 95% noncondensing | | | | | | | | | | | | | |
| Vibration | Operating: 5 Hz to 2k Hz, 0.381 mm (0.015 in.) peak to peak/2.5g panel mounted, ^③ 1hr per axis Non-operating: 5 Hz to 2k Hz, 0.762 mm (0.030 in.) peak to peak/5g, 1hr per axis | | | | | | | | | | | | | |
| Shock ^④ | Operating: 10g peak acceleration (7.5g DIN rail mounted) ^⑤ (11 \pm 1 ms duration) 3 times each direction, each axis Non-operating: 20g peak acceleration (11 \pm 1 ms duration), 3 times each direction, each axis | | | | | | | | | | | | | |
| Agency Certification (when product or packaging is marked) | <ul style="list-style-type: none"> •C-UL Class I, Division 2 Groups A, B, C, D certified •UL listed (Class I, Division 2 Groups A, B, C, D certified) •CE marked for all applicable directives | | | | | | | | | | | | | |
| Terminal Screw Torque | 0.9 N-m maximum (8.0 in.-lbs) | | | | | | | | | | | | | |
| Electrostatic Discharge | IEC801-2 @ 8K V Discrete I/O 4K V Contact, 8K V Air for Analog I/O | | | | | | | | | | | | | |
| Radiated Susceptibility | IEC801-3 @ 10 V/m, 27 MHz – 1000 MHz except for 3V/m, 87 MHz – 108 MHz, 174 MHz – 230 MHz, and 470 MHz – 790 MHz | | | | | | | | | | | | | |
| Fast Transient | IEC801-4 @ 2K V Power Supply, I/O; 1K V Comms | | | | | | | | | | | | | |
| Isolation | 1500V ac | | | | | | | | | | | | | |

^① Refer to page 1–11 for additional information on power supply inrush.

^② DC input voltage derated linearly from 30°C (30V to 26.4V).

^③ DIN rail mounted controller is 1g.

^④ Refer to page 1–14 for vertical mounting specifications.

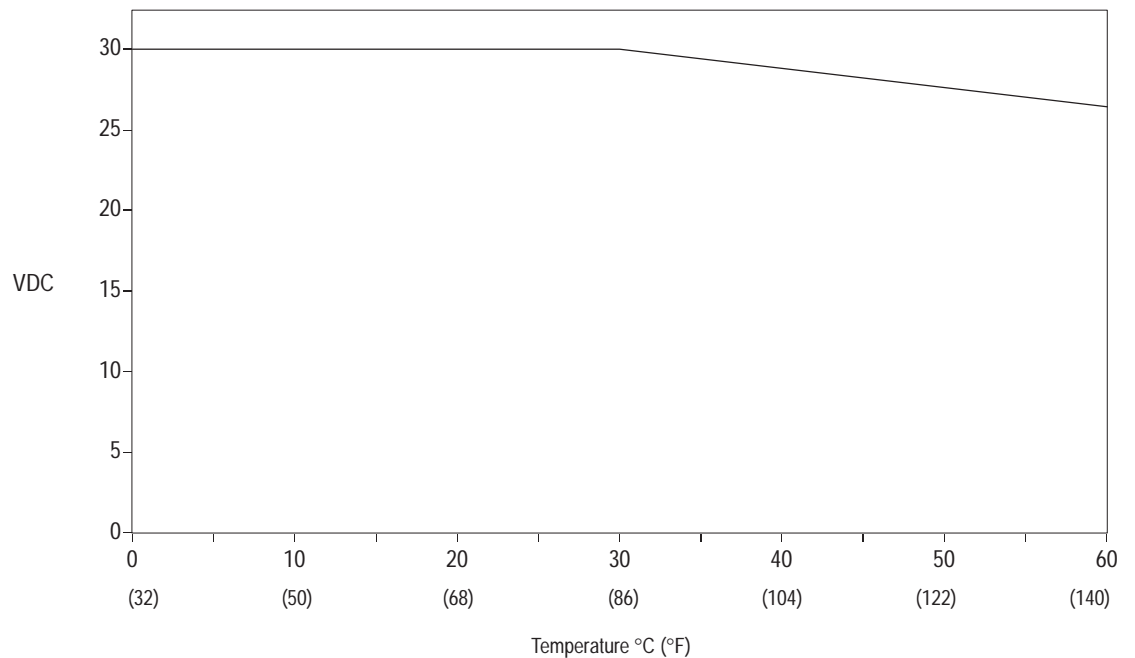
^⑤ Relays are derated an additional 2.5g on 32 pt. controllers.

Input Specifications

| Description | Specification | |
|-------------------|---|--|
| | 100-120V ac Controllers | 24V dc Controllers |
| Voltage Range | 79 to 132V ac 47 to 63 Hz | 14 to 30V dc |
| On Voltage | 79V ac min. 132V ac max. | 14V dc min. 24V dc nominal 26.4V dc max. @ 55°C (131°F) 30.0V dc max. @ 30°C (86°F) |
| Off Voltage | 20V ac | 5V dc |
| On Current | 5.0 mA min. @ 79V ac 47 Hz 12.0 mA nominal @ 120V ac 60 Hz 16.0 mA max. @ 132V ac 63 Hz | 2.5 mA min. @ 14V dc 8.0 mA nominal @ 24V dc 12.0 mA max. @ 30V dc |
| Off Current | 2.5 mA max. | 1.5 mA max. |
| Nominal Impedance | 12K ohms @ 50 Hz 10K ohms @ 60 Hz | 3K ohms |
| Inrush Maximum | 250 mA max. ^① | Not Applicable |

^① To reduce the inrush maximum to 35 mA, apply a 6.8 Kohm, 5w resistor in series with the input. The on-state voltage increases to 92V ac as a result.

dc Input Derating Graph



General Output Specifications

| Type | Relay | MOSFET | Triac |
|-----------------------------------|--|---|---|
| Voltage | See Wiring Diagrams, p. 2–6. | | |
| Maximum Load Current | Refer to the Relay Contact Rating Table. | 1.0A per point @ 55°C (131°F) 1.5A per point @ 30°C (86°F) | 0.5A per point @ 55°C (131°F) 1.0A per point @ 30°C (86°F) |
| Minimum Load Current | 10.0 mA | 1 mA | 10.0 mA |
| Current per Controller | 1440 VA | 3A for L16BBB 6A for L32BBB | 1440 VA |
| Current per Common | 8.0A | 3A for L16BBB 6A for L32BBB | Not Applicable |
| Maximum Off State Leakage Current | 0 mA | 1 mA | 2 mA @ 132V ac 4.5 mA @ 264V ac |
| Off to On Response | 10 ms max. | 0.1 ms | 8.8 ms @ 60 Hz 10.6 ms @ 50 Hz |
| On to Off Response | 10 ms max. | 1 ms | 11.0 ms |
| Surge Current per Point | Not Applicable | 4A for 10 ms ^① | 10A for 25 ms ^① |

^① Repeatability is once every 2 seconds at 55°C (131°F).

Relay Contact Rating Table (applies to all Bulletin 1761 controllers)

| Maximum Volts | Amperes | | Amperes Continuous | Voltamperes | |
|---------------|--------------------|-------|--------------------|-------------|--------|
| | Make | Break | | Make | Break |
| 240V ac | 7.5A | 0.75A | 2.5A | 1800 VA | 180 VA |
| 120V ac | 15A | 1.5A | | | |
| 125V dc | 0.22A ^① | | 1.0A | 28 VA | |
| 24V dc | 1.2A ^① | | 2.0A | 28 VA | |

^① For dc voltage applications, the make/break ampere rating for relay contacts can be determined by dividing 28 VA by the applied dc voltage. For example, $28 \text{ VA} \div 48 \text{ V dc} = 0.58 \text{ A}$. For dc voltage applications less than 48V, the make/break ratings for relay contacts cannot exceed 2A. For dc voltage applications greater than 48V, the make/break ratings for relay contacts cannot exceed 1A.

Analog Input Specifications

| Description | Specification |
|--|--------------------------------------|
| Voltage Input Range | -10.5 to +10.5V dc - 1LSB |
| Current Input Range | -21 to +21 mA - 1LSB |
| Type of Data | 16-bit signed integer |
| Input Coding -21 to +21 mA - 1LSB, -10.5 to +10.5V dc - 1LSB | -32,768 to +32,767 |
| Voltage Input Impedance | 210K Ω |
| Current Input Impedance | 160 Ω |
| Input Resolution ^① | 16 bit |
| Non-linearity | < 0.002% |
| Overall Accuracy 0°C to +55°C | \pm 0.7% of full scale |
| Overall Accuracy Drift 0°C to +55°C (max.) | \pm 0.176% |
| Overall Error at +25°C (+77°F) (max.) | \pm 0.525% |
| Voltage Input Overvoltage Protection | 24V dc |
| Current Input Overcurrent Protection | \pm 50 mA |
| Input to Output Isolation | 30V rated working/500V test 60 Hz/1s |
| Field Wiring to Logic Isolation | |

^① The analog input resolution is a function of the input filter selection.

Analog Output Specifications

| Description | Specification |
|---|----------------------------------|
| Voltage Output Range | 0 to 10V dc -1LSB |
| Current Output Range | 4 to 20 mA - 1LSB |
| Type of Data | 16-bit signed integer |
| Non-linearity | 0.02% |
| Step Response | 2.5 ms (at 95%) |
| Load Range - Voltage Output | 1K Ω to ∞ Ω |
| Load Range - Current Output | 0 to 500 Ω |
| Output Coding 4 to 20 mA - 1 LSB, 0 to 10Vdc - 1LSB | 0 to 32,767 |
| Voltage Output Miswiring | can withstand short circuit |
| Current Output Miswiring | can withstand short circuit |
| Output Resolution | 15 bit |
| Analog Output Settling Time | 3 msec (maximum) |
| Overall Accuracy 0°C to +55°C | \pm 1.0% of full scale |
| Overall Accuracy Drift 0°C to +55°C (max.) | \pm 0.28% |
| Overall Error at +25°C (+77°F) (max.) | 0.2% |
| Field Wiring to Logic Isolation | 30V rated working/500V isolation |

Input Filter Response Times (Discrete)

The input filter response time is the time from when the external input voltage reaches an on or off state to when the micro controller recognizes that change of state. The higher you set the response time, the longer it takes for the input state change to reach the micro controller. However, setting higher response times also provides better filtering of high frequency noise.

You can apply a unique input filter setting to each of the three input groups:

- 0 and 1
- 2 and 3
- 4 to x; where x=9 for 16 I/O point controllers, and x=19 for 32 I/O point controllers

The minimum and maximum response times associated with each input filter setting can be found in the tables that follow.

Response Times for High-Speed dc Inputs 0 to 3 (applies to 1761-L10BWA, 1761-L16BWA, -L20BWA-5A, -L32BWA, -L10BWB, -L16BWB, -L20BWB-5A, -L32BWB, -L16BBB, and -L32BBB controllers)

| Maximum High-Speed Counter Frequency @ 50% Duty Cycle (Khz) | Nominal Filter Setting (ms) | Maximum ON Delay (ms) | Maximum OFF Delay (ms) |
|---|-----------------------------|-----------------------|------------------------|
| 6.600 | 0.075 | 0.075 | 0.075 |
| 5.000 | 0.100 | 0.100 | 0.100 |
| 2.000 | 0.250 | 0.250 | 0.250 |
| 1.000 | 0.500 | 0.500 | 0.500 |
| 0.500 | 1.000 | 1.000 | 1.000 |
| 0.200 | 2.000 | 2.000 | 2.000 |
| 0.125 | 4.000 | 4.000 | 4.000 |
| 0.062 | 8.000 ^① | 8.000 | 8.000 |
| 0.031 | 16.000 | 16.000 | 16.000 |

^① This is the default setting.

Response Times for dc Inputs 4 and Above (applies to 1761-L10BWA, 1761-L16BWA, -L20BWA-5A, -L32BWA, -L10BWB, -L16BWB, -L20BWB-5A, -L32BWB, -L16BBB, and -L32BBB controllers)

| Nominal Filter Setting (ms) | Maximum ON Delay (ms) | Maximum OFF Delay (ms) |
|-----------------------------|-----------------------|------------------------|
| 0.50 | 0.500 | 0.500 |
| 1.00 | 1.00 | 1.000 |
| 2.00 | 2.000 | 2.000 |
| 4.00 | 4.000 | 4.000 |
| 8.00 ^① | 8.000 | 8.000 |
| 16.00 | 16.000 | 16.000 |

^① This is the default setting.

Response Times for ac Inputs (applies to 1761-L16AWA, -L20AWA-5A, -L32AWA, and -L32AAA controllers)

| Nominal Filter Setting (ms) ^① | Maximum ON Delay (ms) | Maximum OFF Delay (ms) |
|--|-----------------------|------------------------|
| 8.0 | 20.0 | 20.0 |

^① There is only one filter setting available for the ac inputs. If you make another selection, the controller changes it to the ac setting and sets the input filter modified bit (S:5/13).

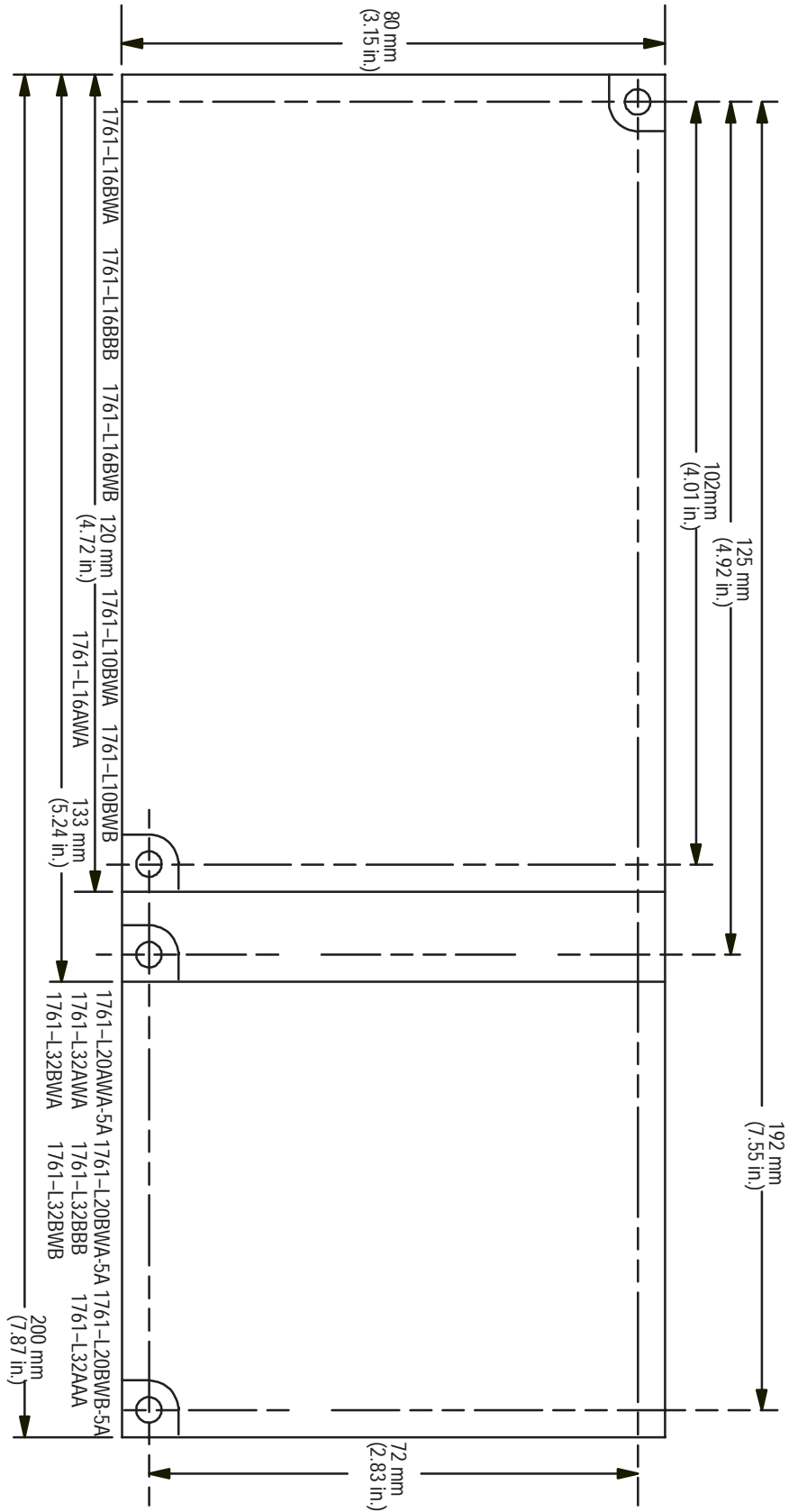
Controller Dimensions

Refer to the following table for the controller dimensions.

| Controller: 1761- | Length: mm (in.) | Depth: mm (in.) ^① | Height: mm (in.) |
|-------------------|------------------|------------------------------|------------------|
| L10BWA | 120 (4.72) | 73 (2.87) | 80 (3.15) |
| L16AWA | 133 (5.24) | | |
| L16BWA | 120 (4.72) | | |
| L20AWA-5A | 200 (7.87) | | |
| L20BWA-5A | | | |
| L32AWA | | | |
| L32BWA | | | |
| L32AAA | | | |
| L10BWB | 120 (4.72) | 40 (1.57) | |
| L16BBB | | | |
| L16BWB | | | |
| L20BWB-5A | 200 (7.87) | | |
| L32BBB | | | |
| L32BWB | | | |

^① Add approximately 13 mm (0.51 in.) when using the 1761-CBL-PM02 or 1761-CBL-HM02 communication cables.

Copy the template on the following page to help you install your controller.



Hand-Held Programmer Specifications

The following tables summarize the specifications and dimensions for the MicroLogix 1000 HHP.

General Specifications

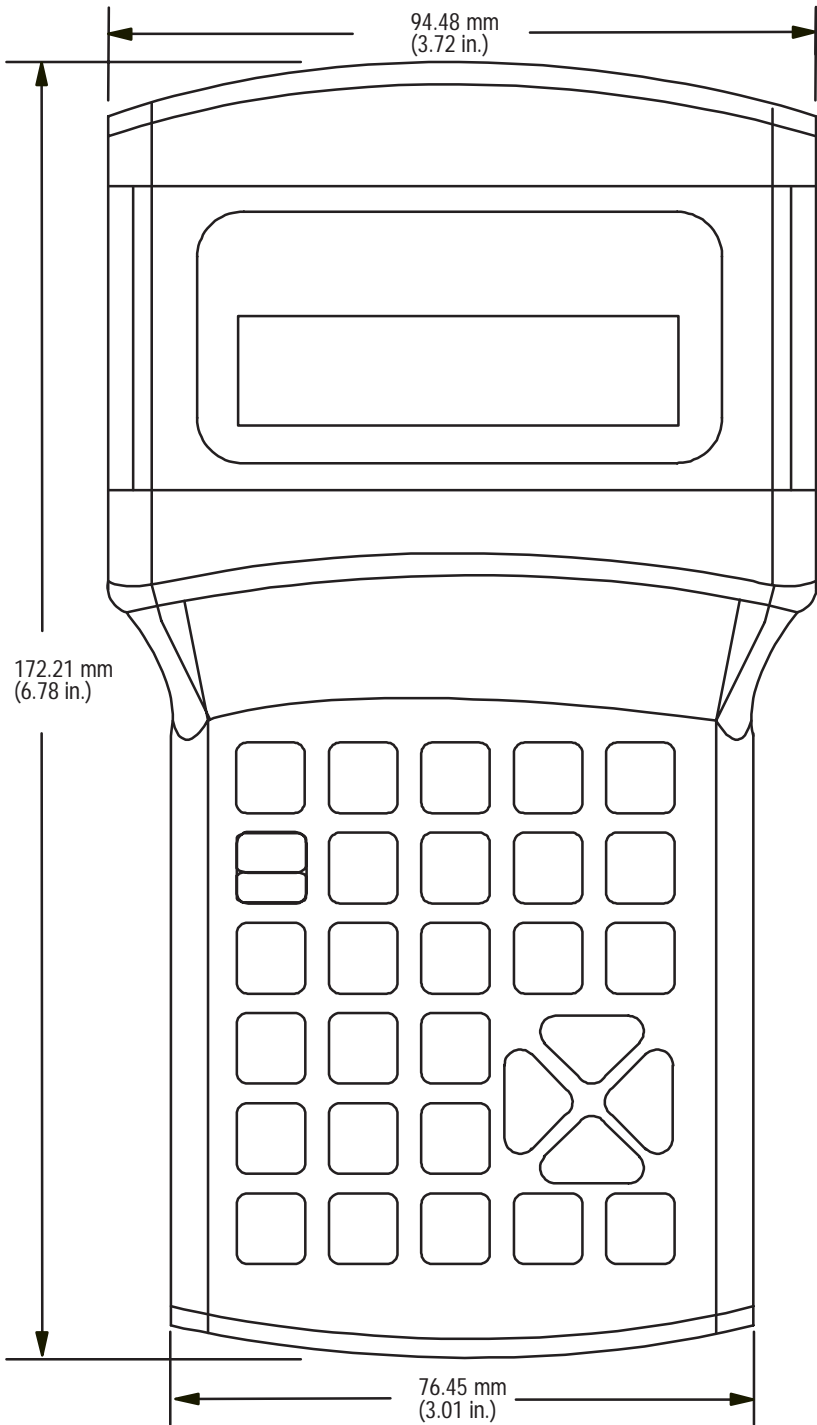
| Description | Specification: 1761-HHP-B30 |
|--|---|
| Operating Power | 2.0W |
| Operating Temperature | 0°C to +50°C (+32°F to +122°F) |
| Storage Temperature | -20°C to +60°C (-4°F to +140°F) |
| Operating Humidity | 5 to 95% noncondensing |
| Vibration (Operating and Non-operating) | 10 to 500 Hz 0.762 mm (0.030 in.) peak to peak 1.5g maximum peak acceleration 1 hr in x, y, and z axis |
| Shock | Operating: 50g peak acceleration for 11±1 ms duration 3 times in x, y, and z axis |
| | Non-operating: 50g peak acceleration for 11±1 ms duration 3 times in x, y, and z axis |
| Agency Certification (when product or packaging is marked) | <ul style="list-style-type: none"> •C-UL Class 1, Division 2 Groups A, B, C, D certified •UL listed (Class 1, Division 2 Groups A, B, C, D certified) •CE marked for all applicable directives |
| Display Type | LCD 16 character x 2 line display |
| Keypad | rubber/carbon colored coded |

Input Specifications

| Description | Specification |
|-------------|---------------|
| Type | 24V dc |

Dimensions

| Terminal: 1761- | Width: mm (in.) | Height: mm (in.) | Depth: mm (in.) |
|-----------------|--------------------|---------------------|--------------------|
| HHP | 95 (3.74) | 170 (6.69) | 35 (1.37) |



Controller and Hand-Held Programmer Accessories and Replacement Parts

This table provides a list of accessories and replacement parts and their catalog numbers.

| Description | Catalog Number |
|---|----------------|
| 10 pt. ac input, 6 pt. relay output, ac power supply controller | 1761-L16AWA |
| 12 pt. ac and 4 pt. analog inputs, 8 pt. relay and 1 pt. analog outputs, ac power supply controller | 1761-L20AWA-5A |
| 20 pt. ac input, 12 pt. relay output, ac power supply controller | 1761-L32AWA |
| 6 pt. dc input, 4 pt. relay output, ac power supply controller | 1761-L10BWA |
| 10 pt. dc input, 6 pt. relay output, ac power supply controller | 1761-L16BWA |
| 12 pt. dc and 4 pt. analog inputs, 8 pt. relay and 1 pt. analog outputs, ac power supply controller | 1761-L20BWA-5A |
| 20 pt. dc input, 12 pt. relay output, ac power supply controller | 1761-L32BWA |
| 6 pt. dc input, 4 pt. relay output, dc power supply controller | 1761-L10BWB |
| 10 pt. dc input, 6 pt. relay output, dc power supply controller | 1761-L16BWB |
| 12 pt. dc and 4 pt. analog inputs, 8 pt. relay and 1 pt. analog outputs, dc power supply controller | 1761-L20BWB-5A |
| 20 pt. dc input, 12 pt. relay output, dc power supply controller | 1761-L32BWB |
| 10 pt. dc input, 4 pt. FET and 2 pt. relay outputs, dc power supply controller | 1761-L16BBB |
| 20 pt. dc input, 10 pt. FET and 2 pt. relay outputs, dc power supply controller | 1761-L32BBB |
| 20 pt. ac input, 10 pt. triac and 2 pt. relay outputs, ac power supply controller | 1761-L32AAA |
| Terminal doors for -L16AWA (2 doors per package) | 1761-RPL-T16A |
| Terminal doors for -L16BWA (2 doors per package) | 1761-RPL-T16B |
| Terminal doors for -L32AWA, -L32BWA, or -L32AAA (2 doors per package) | 1761-RPL-T32X |
| Communications door (1 door per package) | 1761-RPL-COM |
| DIN rail latches (2 per package) | 1761-RPL-DIN |
| 2.00 m (6.56 ft) cable (DIN-to-DIN) for use with the MicroLogix 1000 HHP | 1761-CBL-HM02 |
| Hand-Held Programmer (includes 1761-CBL-HM02 communication cable) | 1761-HHP-B30 |
| Memory module for 1761-HHP-B30 (stores 1 program) | 1761-HHM-K08 |
| Memory module for 1761-HHP-B30 (stores 8 programs) | 1761-HHM-K64 |
| Memory module door for 1761-HHP-B30 (1 door per package) | 1761-RPL-TRM |

Programming Reference

This appendix lists the:

- controller status file
- function codes
- instruction execution times and instruction memory usage

Controller Status File

The status file lets you monitor how your operating system works and lets you direct how you want it to work. This is done by using the status file to set up control bits and monitor both hardware and software faults and other status information.

Important: Do not write to reserved words in the status file. If you intend writing to status file data, it is imperative that you first understand the function fully.

The status file (S) contains the following words:

| Word | Function | Page |
|-------------------|--|------|
| S:0 | Arithmetic Flags | B-2 |
| S:1L (low byte) | Controller Mode Status/Control (low) | B-3 |
| S:1H (high byte) | Controller Mode Status/Control (Hi) | B-3 |
| S:2L (low byte) | Controller Alternate Mode Status/Control (low) | B-5 |
| S:2H (high byte) | Controller Alternate Mode Status/Control (Hi) | B-5 |
| S:3L (low byte) | Current Scan Time | B-6 |
| S:3H (high byte) | Watchdog Scan Time | B-7 |
| S:4 | Timebase | B-7 |
| S:5 | Minor Error Bits | B-7 |
| S:6 | Major Error Code | B-8 |
| S:7 | Suspend Code | B-11 |
| S:8 to S:12 | Reserved | B-11 |
| S:13, S:14 | Math Register | B-11 |
| S:15L (low byte) | DF1 Full or Half-Duplex Node Address | B-11 |
| S:15H (high byte) | DF1 Full or Half-Duplex Baud Rate | B-11 |
| S:16L (low byte) | DH-485 Node Address | B-11 |
| S:16H (high byte) | DH-485 Baud Rate | B-12 |
| S:17 to S:21 | Reserved | B-12 |
| S:22 | Maximum Observed Scan Time | B-12 |

Continued on next page

| Word | Function | Page |
|---------------|----------------|------|
| S:23 | Reserved | B-12 |
| S:24 | Index Register | B-12 |
| S:25 to S:29 | Reserved | B-12 |
| S:30 | STI Setpoint | B-12 |
| S:31 and S:32 | Reserved | B-12 |

Status File Descriptions

The following tables describe the status file functions, beginning at address S0 and ending at address S32.

Each status bit is classified as one of the following:

- **Status** — Use these words, bytes, or bits to monitor controller operation or controller status information. The information is seldom written to by the user program or programming device (unless you want to reset or clear a function such as a monitor bit).
- **Dynamic Configuration** — Use these words, bytes, or bits to select controller options while online with the controller.
- **Static Configuration** — Use these words, bytes, or bits to select controller options prior to saving the user program.


| Address | Bit | Classification | Description |
|--------------|----------------------------------|----------------|---|
| S0 | Arithmetic and Scan Status Flags | | The arithmetic flags are assessed by the controller following the execution of certain math and data handling instructions. The state of these bits remain in effect until certain math or data handling instructions in the program are executed. |
| S0/0 | Carry | Status | This bit is set by the controller if a mathematical carry or borrow is generated. Otherwise the bit remains cleared. This bit is assessed as if a function of unsigned math. When a STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of S:0/0 is restored when execution resumes. |
| S0/1 | Overflow | Status | This bit is set by the controller when the result of a mathematical operation does not fit in its destination. Otherwise the bit remains cleared. Whenever this bit is set, the overflow trap bit S:5/0 is also set except for the ENC bit. Refer to S:5/0. When a STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of S:0/1 is restored when execution resumes. |
| S0/2 | Zero | Status | This bit is set by the controller when the result of certain math or data handling instructions is zero. Otherwise the bit remains cleared. When a STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of S:0/2 is restored when execution resumes. |
| S0/3 | Sign | Status | This bit is set by the controller when the result of certain math or data handling instructions is negative. Otherwise the bit remains cleared. When a STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of S:0/3 is restored when execution resumes. |
| S0/4 to S0/7 | Reserved | NA | NA |

| Address | Bit | Classification | Description |
|---------------------------|---------------------------------|----------------------|--|
| S0/8 ^① | Extend I/O Configuration | Static Configuration | This bit must be set by the user when unused outputs are written to. If reset and unused outputs are turned on the controller will fault (41H). |
| S0/9 | Reserved | NA | NA |
| S0/10 | Primary Protocol | Static Configuration | This bit defines the protocol that the controller initially uses when attempting to establish communication, where: 0 = DF1 (default setting) 1 = DH-485 |
| S0/11 | Active Protocol | Status | This bit is updated by the controller during a protocol switch. It indicates which protocol is currently being used for communication, where: 0 = DF1 1 = DH-485 |
| S0/12 ^② | Selected DF1 Protocol | Status | This bit allows the user to determine which DF1 protocol is configured, where: 0 = DF1 Full-Duplex (default setting) 1 = DF1 Half-Duplex Slave |
| S0/13 to S0/15 | Reserved | NA | NA |
| S1/0 to S1/4 ^③ | Controller Mode Status/ Control | Status | Bits 0–4 function as follows: 0 0000 = (0) Remote Download in progress 0 0001 = (1) Remote Program mode 0 0011 = (3) Suspend Idle (operation halted by SUS instruction execution) 0 0110 = (6) Remote Run mode 0 0111 = (7) Remote Test continuous mode 0 1000 = (8) Remote Test single scan mode |
| S1/5 ^③ | Forces Enabled | Status | This bit is set by the controller (1) to indicate that forces are always enabled. |
| S1/6 ^③ | Forces Installed | Status | This bit is set by the controller to indicate that forces have been set by the user. |
| S1/7 ^③ | Comms Active | Status | This bit is set when the controller receives valid data from the communication port. For DF1 protocols, the bit is reset if the controller does not receive valid data from the programming port for 10 seconds. Note: In DF1 half-duplex mode, simple polls by the DF1 master or replies to received messages will not reset the timer. A poll with a command is required to reset the timer. For DH-485, the bit is reset as soon as the DH-485 link layer determines that no other devices are active on the link. Application Note: For DF1 half-duplex, you can use this bit to enable a timer (via an XIO instruction) to sense whether the DF1 master is actively communicating to the slave. The preset of the timer is determined by the total network timing. |
| S1/8 ^③ | Fault Override at Powerup | Static Configuration | When set, this bit causes the controller to clear the Major Error Halted bit S:1/13 and Minor error bits S:5/0 to S:5/7 on power up if the processor had previously been in the REM Run mode and had faulted. The controller then attempts to enter the REM Run mode. Set this bit offline only. |

^① Valid for Series A–C discrete only.

^② This bit is set in the COMMS menu. It is not displayed in the status file of the HHP.

^③ Address is not shown in HHP data monitor.

| Address | Bit | Classification | Description |
|--------------------|--------------------------|-----------------------|--|
| S1/9 ^① | Startup Protection Fault | Static Configuration | <p>When this bit is set and power is cycled when the controller powers up in Run mode, the controller executes the user-fault routine prior to the execution of the first scan of your program. You have the option of clearing the Major Error Halted bit S:1/13 to resume operation in the REM Run mode. If the user-fault routine does not reset bit S:1/13, the fault mode results.</p> <p>Program the user-fault routine logic accordingly. When executing the startup protection fault routine, S:6 (major error fault code) contains the value 0016H.</p> |
| S1/10 to S1/11 | Reserved | NA | NA |
| S1/12 ^① | Run Always | Static Configuration | <p>When set, this bit causes the controller to clear S:1/13 before attempting to enter RUN mode when power is applied or if an unexpected reset occurs. If this bit is not set, the controller powers up in the previous mode it was in before losing power, unless the controller was in REM test mode. If the controller was in REM test mode when power was removed, the controller enters REM program mode when power is applied.</p> <p>This bit overrides any faults existing at power down.</p> <p> ATTENTION: Setting the Run Always bit causes the controller to enter the REM RUN mode if an unexpected reset occurs, regardless of the mode that the controller was in before the reset occurred. Unexpected resets may occur due to electromagnetic noise, improper grounding, or an internal controller hardware failure. Make sure your application is designed to safely handle this situation.</p> |
| S1/13 | Major Error Halted | Dynamic Configuration | <p>This bit is set by the controller any time a major error is encountered. The controller enters a fault condition. Word S:6, the Fault Code contains a code that can be used to diagnose the fault condition. Any time bit S:1/13 is set, the controller:</p> <ul style="list-style-type: none"> • either places all outputs in a safe state (outputs are off) and energizes the fault LED (blinking), • or enters the user-fault routine with outputs active (if in REM Run mode), allowing the fault routine ladder logic to attempt recovery from the fault condition. If the user-fault routine determines that recovery is required, clear S:1/13 using ladder logic prior to exiting the fault routine. If the fault routine ladder logic does not understand the fault code, or if the routine determines that it is not desirable to continue operation, the controller exits the fault routine with bit S:1/13 set. The outputs are placed in a safe state and the FAULT LED is blinking. |

^① Address is not shown in HHP data monitor.

Continued on next page

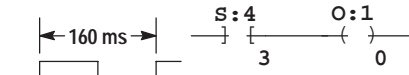
| Address | Bit | Classification | Description ¹ |
|-------------------|------------------------------|---------------------------------|---|
| S1/13 | Major Error Halted | Dynamic Configuration | When you clear bit S:1/13 using a programming device, the controller mode changes from fault to Remote Program. You can move a value to S:6, then set S:1/13 in your ladder program to generate an application specific major error. All application generated faults are recoverable regardless of the value used. Note: <i>Once a major fault state exists, you must correct the condition causing the fault, and you must also clear this bit in order for the controller to accept a mode change attempt (into REM Run or REM Test). Also, clear S:6 to avoid the confusion of having an error code but no fault condition.</i> |
| S1/13 | Major Error Halted | Dynamic Configuration | Note: <i>Do not re-use error codes that are defined later in this appendix as application specific error codes. Instead, create your own unique codes. This prevents you from confusing application errors with system errors. We recommend using error codes FFOO to FFOF to indicate application specific major errors.</i> |
| S1/14 | OEM Lock | Static Configuration | Using this bit you can control access to a controller file. To program this feature, select "Future Access Disallow" when saving your program. When this bit is cleared, it indicates that any compatible programming device can access the ladder program (provided that password conditions are satisfied). |
| S1/15 | First Pass | Status | Use this bit to initialize your program as the application requires. When this bit is set by the controller, it indicates that the first scan of the user program is in progress (following power up in the RUN mode or entry into a REM Run or REM Test mode). The controller clears this bit following the first scan. This bit is set during execution of the startup protection fault routine. Refer to S:1/9 for more information. |
| S2/0 | STI Pending | Status | When set, this bit indicates that the STI timer has timed out and the STI routine is waiting to be executed. This bit is cleared upon starting the STI routine, ladder program, exit of the REM Run or Test mode, or execution of a true STS instruction. |
| S2/1 | STI Enabled | Status and Static Configuration | This bit may be set or reset using the STS, STE, or STD instruction. If set, it allows execution of the STI if the STI setpoint S:30 is non-zero. If clear, when an interrupt occurs, the STI subroutine does not execute and the STI Pending bit is set. The STI Timer continues to run when this bit is disabled. The STD instruction clears this bit. If this bit is set or reset editing the status file online, the STI is not affected. If this bit is set, the bit allows execution of the STI. If this bit is reset editing the status file offline, the bit disallows execution of the STI. |
| S2/2 | STI Executing | Status | When set, this bit indicates that the STI timer has timed out and the STI subroutine is currently being executed. This bit is cleared upon completion of the STI routine, ladder program, or REM Run or Test mode. |
| S2/3 to S2/4 | Reserved | NA | NA |
| S2/5 ^① | Incoming Command Pending Bit | Status | This bit is set when the processor determines that another node on the network has requested information or supplied a command to it. This bit can be set at any time. This bit is cleared when the processor services the request (or command). |

^① Valid for Series C discrete only.

Continued on next page

| | | | |
|-------------------|--------------------------------------|-----------------------|--|
| S2/6 ^① | Message Reply Pending Bit | Status | This bit is set when another node on the network has supplied the information you requested in the MSG instruction of your processor. This bit is cleared when the processor stores the information and updates your MSG instruction. |
| S2/7 ^① | Outgoing Message Command Pending Bit | Status | This bit is set when one or more messages in your program are enabled and waiting, but no message is being transmitted at the time. As soon as transmission of a message begins, the bit is cleared. After transmission, the bit is set again if there are further messages waiting. It remains cleared if there are no further messages waiting. |
| S2/8 to S2/13 | Reserved | NA | NA |
| S2/14 | Math Overflow Selection | Dynamic Configuration | <p>Set this bit when you intend to use 32-bit addition and subtraction. When S:2/14 is set, and the result of an ADD, SUB, MUL, or DIV instruction cannot be represented in the destination address (underflow or overflow):</p> <ul style="list-style-type: none"> the overflow bit S:0/1 is set, the overflow trap bit S:5/0 is set, and the destination address contains the unsigned truncated least significant 16 bits of the result. <p>The default condition of S:2/14 is reset (0). When S:2/14 is reset, and the result of an ADD, SUB, MUL, or DIV instruction cannot be represented in the destination address (underflow or overflow):</p> <ul style="list-style-type: none"> the overflow bit S:0/1 is set, the overflow trap bit S:5/0 is set, and the destination address contains 32767 if the result is positive or – 32768 if the result is negative. <p>Note that the status of bit S:2/14 has no effect on the DDV instruction. Also, it has no effect on the math register content when using MUL and DIV instructions.</p> <p>To provide protection from inadvertent alteration of your selection, program an unconditional OTL instruction at address S:2/14 to ensure the new math overflow operation. Program an unconditional OTU instruction at address S:2/14 to ensure the original math overflow operation.</p> |
| S2/15 | Reserved | NA | NA |
| S3L | Current Scan Time | Status | <p>The value of this byte tells you how much time elapses in a program cycle. A program cycle includes:</p> <ul style="list-style-type: none"> scanning the ladder program, housekeeping, scanning the I/O, servicing of the communication channel. <p>The byte value is zeroed by the controller each scan, immediately preceding the execution of rung 0 of program file 2 (main program file). The byte is incremented every 10 ms thereafter and indicates, in 10 ms increments, the amount of time elapsed in each scan. If this value ever equals the value in S:3H Watchdog, a user watchdog major error is declared (code 0022).</p> <p>The resolution of the scan time value is +0 to –10 ms. Example: The value 9 indicates that 80–90 ms has elapsed since the start of the program cycle.</p> |

^① Valid for Series C discrete only.

| Address | Bit | Classification | Description |
|------------------|------------------------|-----------------------|---|
| S3H ^① | Watchdog Scan Time | Dynamic Configuration | This byte value contains the number of 10 ms ticks allowed to occur during a program cycle. The default value is 10 (100 ms), but you can increase this to 255 (2.55 seconds) or decrease it to 1, as your application requires. If the program scan S:3L value equals the watchdog value, a watchdog major error is declared (code 0022). |
| S4 | Timebase | Status | <p>All 16 bits of this word are assessed by the controller. The value of this word is zeroed upon power up in the REM Run mode or entry into the REM Run or REM Test mode. It is incremented every 10 ms thereafter.</p> <p>Application note: You can write any value to S:4. It will begin incrementing from this value.</p> <p>You can use any individual bit of this word in your user program as a 50% duty cycle clock bit. Clock rates for S:4/0 to S:4/15 are: 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10240, 20480, 40960, 81920, 163840, 327680, and 655360 ms.</p> <p>The application using the bit must be evaluated at a rate more than two times faster than the clock rate of the bit. In the example below, bit S:4/3 toggles every 80 ms, producing a 160 ms clock rate. To maintain accuracy of this bit in your application, the instruction using bit S:4/3 (O:1/0 in this case) must be evaluated at least once every 79.999 ms</p>  <p>Both S:4/3 and Output O:1/0 toggle every 80 ms. O:1/0 must be evaluated at least once every 79.999 ms.</p> |
| S5 | Minor Error Bits | | The bits of this word are set by the controller to indicate that a minor error has occurred in your ladder program. Minor errors, bits 0 to 7, revert to major error 0020H if any bit is detected as being set at the end of the scan. These bits are automatically cleared on a power cycle. |
| S5/0 | Overflow Trap | Dynamic Configuration | <p>When this bit is set by the controller, it indicates that a mathematical overflow has occurred in the ladder program. See S:0/1 for more information.</p> <p>If this bit is ever set upon execution of the END or TND instruction, major error (0020) is declared. To avoid this type of major error from occurring, examine the state of this bit following a math instruction (ADD, SUB, MUL, DIV, DDV, NEG, SCL, TOD, or FRD), take appropriate action, and then clear bit S:5/0 using an OTU instruction with S:5/0.</p> |
| S5/1 | Reserved | NA | NA |
| S5/2 | Control Register Error | Dynamic Configuration | The LFU, LFL, FFU, FFL, BSL, BSR, SQO, SQC, and SQL instructions are capable of generating this error. When bit S:5/2 is set, it indicates that the error bit of a control word used by the instruction has been set. |

^① Address is not shown in HHP data monitor.

| Address | Bit | Classification | Description |
|----------------|---|-----------------------|--|
| S5/2 | Control Register Error | Dynamic Configuration | If this bit is ever set upon execution of the END or TND instruction, major error (0020) is declared. To avoid this type of major error from occurring, examine the state of this bit following a control register instruction, take appropriate action, and then clear bit S:5/2 using an OTU instruction with S:5/2. |
| S5/3 | Major Error Detected While Executing user-fault routine | Dynamic Configuration | When set, the major error code (S:6) represents the major error that occurred while processing the fault routine due to another major error. |
| S5/4 to S5/7 | Reserved | NA | NA |
| S5/8 | Retentive Data Lost | Status | This bit is set whenever retentive data is lost. This bit remains set until you clear it. While set, this bit causes the controller to fault prior to the first true scan of the program. |
| S5/9 | Reserved | NA | NA |
| S5/10 | STI Lost | Status | This bit is set whenever the STI timer expires while the STI routine is either executing or disabled <i>and</i> the pending bit (S:2/0) is already set. |
| S5/11 to S5/12 | Reserved | NA | NA |
| S5/13 | Input Filter Selection Modified | Status | This bit is set whenever the discrete input filter selection in the controller is made compatible with the hardware. Refer to page A-6 for more information. |
| S5/14 to S5/15 | Reserved | NA | NA |
| S6 | Major Error Code | Status | <p>A hexadecimal code is entered in this word by the controller when a major error is declared. Refer to S:1/13. The code defines the type of fault, as indicated on the following pages. This word is not cleared by the controller.</p> <p>Error codes are presented, stored, and displayed in a hexadecimal format.</p> <p>If you enter a fault code as a parameter in an instruction in your ladder program, you must convert the code to decimal.</p> <p>Application note: You can declare your own application specific major fault by writing a unique value to S:6 and then setting bit S:1/13. Interrogate the value of S:6 in the user-fault routine to determine the type of fault that occurred.</p> <p>Fault Classifications: Faults are classified as Non-User, Non-Recoverable, and Recoverable.</p> <p>Error code descriptions and classifications are listed on the following pages. Categories are:</p> <ul style="list-style-type: none"> ● power up errors ● going-to-run errors ● run errors ● download errors |

NA = Not Applicable

Each fault is classified as one of the following:

- Non-User — A fault caused by various conditions that cease logic program execution. The user-fault routine is not run when this fault occurs.
- Non-Recoverable — A fault caused by the user that cannot be recovered from. The user-fault routine is run when this fault occurs. However, the fault cannot be cleared.
- Recoverable — A fault caused by the user that can be recovered from in the user-fault routine by resetting major error halted bit (S1/13). The user-fault routine is run when this fault occurs.

Refer to chapter 20 for more information regarding MicroLogix 1000 HHP advisory messages.

| Address | Error Code (Hex) | <i>Power up Errors</i> | Fault Classification | | |
|---------|------------------|---|----------------------|-----------------|-------------|
| | | | Non-User | User | |
| | | | | Non-Recoverable | Recoverable |
| S6 | 0001 | The default program was loaded. | X | | |
| | 0002 | Unexpected reset occurred. | X | | |
| | 0003 | EEPROM memory is corrupt. | X | | |
| | 0008 | A fatal internal programming device error occurred. | X | | |
| | 0009 | A fatal internal hardware error occurred. | X | | |

| Address | Error Code (Hex) | <i>Going-to-Run (GTR) Errors^①</i> | Fault Classification | | |
|---------|------------------|--|----------------------|-----------------|-------------|
| | | | Non-User | User | |
| | | | | Non-Recoverable | Recoverable |
| S6 | 0005 | Retentive data is lost. | | | X |
| | 0010 | The downloaded program is not a controller program. | X | | |
| | 0016 | Startup protection after power loss, S:1/9 is set. The user must check for a retentive data lost condition if the user-fault routine was executed with startup protection. | | | X |

^① Going-to-Run errors occur when the controller is going from any mode to REM Run mode or from any non-Run mode (PRG, SUS) to Test mode.

| Address | Error Code (Hex) | <i>Run Errors</i> | Fault Classification | | |
|-------------------|----------------------------------|---|----------------------|-----------------|-------------|
| | | | Non-User | User | |
| | | | | Non-Recoverable | Recoverable |
| S6 | 0004 | A runtime memory integrity error occurred. | X | | |
| | 0020 | A minor error at the end of the scan. Refer to S:5. | | | X |
| | 0022 | The watchdog timer expired. Refer to S:3H. | | X | |
| | 0024 | Invalid STI interrupt setpoint. Refer to S:30. | | X | |
| | 0025 | There are excessive JSRs in the STI subroutine (file 5). | | X | |
| | 0027 | There are excessive JSRs in the fault subroutine (file 3). | | X | |
| | 002A | The indexed address is too large for the file. | | X | |
| | 002B | There are excessive JSRs in the high-speed counter subroutine (file 4). | | X | |
| | 0030 | The subroutine nesting exceeds a limit of 8 (file 2). | | X | |
| | 0031 | An unsupported instruction was detected. | X | | |
| | 0032 | An SQO/SQC instruction crossed data file boundaries. | | | X |
| | 0033 | The LFU, LFL, FFU, FFL, BSL, or BSR instruction crossed data file boundaries. | | | X |
| | 0034 | A negative value for a timer accumulator or preset value was detected. | | | X |
| | 0035 | An illegal instruction (TND) occurred in the interrupt file. | | X | |
| | 0037 | Invalid presets were loaded to the high-speed counter. | | | X |
| | 0038 | A RET instruction was detected in program file 2. | X | | |
| 0040 | An output verify write occurred. | | X | | |
| 0041 ^① | Extra output bit(s) turned on. | | X | | |

^① Valid for Series A–C discrete only.

| Address | Error Code (Hex) | <i>Saving to Controller or Loading from Memory Module Errors</i> | Fault Classification | | |
|---------|------------------|--|----------------------|-----------------|-------------|
| | | | Non-User | User | |
| | | | | Non-Recoverable | Recoverable |
| S6 | 0018 | The user program is incompatible with the operating system. | X | | |

| Address | Bit | Classification | Description |
|-------------------|---------------------|----------------|---|
| S7 | Suspend Code | Status | <p>When a non-zero value appears in S:7, it indicates that the SUS instruction identified by this value has been evaluated as true and the Suspend Idle mode is in effect. This pinpoints the conditions in the application that caused the Suspend Idle mode. This value is not cleared by the controller.</p> <p>Use the SUS instruction with startup troubleshooting, or as runtime diagnostics for detection of system errors.</p> |
| S8 to S12 | Reserved | NA | NA |
| S13 and S14 | Math Register | Status | <p>Use this double register to produce 32-bit signed divide and multiply operations, precision divide or double divide operations, and 5-digit BCD conversions.</p> <p>These two words are used in conjunction with the MUL, DIV, DDV, FRD, and TOD math instructions. The math register value is assessed upon execution of the instruction and remains valid until the next MUL, DIV, DDV, FRD, or TOD instruction is executed in the user program.</p> <p>An explanation of how the math register operates is included with the instruction definitions.</p> <p>If you store 32-bit signed data values, you must manage this data type without the aid of an assigned 32-bit data type. For example, combine B3:0 and B3:1 to create a 32-bit signed data value. We recommend that you start all 32-bit values on an even or odd word boundary for ease of application and viewing. Also, we recommend that you design, document, and view the contents of 32-bit signed data in either the hexadecimal or binary radix.</p> <p>When an STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of the math register is restored when execution resumes.</p> |
| S15L ^① | DF1 Node Address | Status | <p>This byte value contains the node address of your processor on the DF1 link. It is used when executing Message (MSG) instructions over the DF1 link. The default node address of a processor is 1. Valid node addresses are 0–254. To change a processor node address you must use a programming device.</p> |
| S15H ^① | DF1 Baud Rate | Status | <p>This byte value contains a code used to select the baud rate of the processor on the DF1 link.</p> <p>The controller baud rate options are:</p> <ul style="list-style-type: none"> ● 300 ● 600 ● 1200 ● 2400 ● 4800 (MicroLogix 1000 Series D or later discrete and all MicroLogix 1000 analog only) ● 9600 (default) ● 19200 ● 38400 (MicroLogix 1000 Series D or later discrete and all MicroLogix 1000 analog only) <p>To change the baud rate from the default value you must use a programming device.</p> |
| S16L ^① | DH-485 Node Address | Status | <p>This byte value contains the node address of your processor on the DH-485 link. Each device on the DH-485 link must have a unique address between the decimal values 1–31. To change a processor node address, you must use a programming device.</p> |

^① Address is not shown in HHP data monitor.

| Address | Bit | Classification | Description |
|-------------------|----------------------------|-----------------------|---|
| S16H ^① | DH-485 Baud Rate | Status | This byte value contains the baud rate of the processor on the DH-485 link. The controller baud rate options are: <ul style="list-style-type: none"> ● 9600 ● 19200 (default) To change the baud rate from the default value, you must use a programming device. |
| S17 to S21 | Reserved | NA | NA |
| S22 | Maximum Observed Scan Time | Dynamic Configuration | This word indicates the maximum observed interval between consecutive program cycles. This value indicates, in 10 ms increments, the time elapsed in the longest program cycle of the controller. Refer to S:3L for more information regarding the program cycle. The controller compares each last scan value to the value contained in S:22. If the controller determines that the last scan value is larger than the value stored at S:22, the last scan value is written to S:22. Resolution of the maximum observed scan time value is +0 to -10 ms. For example, the value 9 indicates that 80-90 ms was observed as the longest program cycle. Interrogate this value if you need to determine or verify the longest scan time of your program. |
| S23 | Reserved | NA | NA |
| S24 | Index Register | Status | This word indicates the element offset used in indexed addressing. When an STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of this register is restored when execution resumes. |
| S25 to S29 | Reserved | NA | NA |
| S30 ^① | STI Setpoint | Dynamic Configuration | You enter the timebase to be used in the selectable timed interrupt (STI). The time can range from 10 to 2550 ms. (This is in 10ms increments, so valid values are from 0-255.) Your STI routine executes per the value you enter. Write a zero value to disable the STI. To provide protection from inadvertent alteration of your selection, program an unconditional MOV instruction containing the setpoint value of your STI to S:30, or program a CLR instruction at S:30 to prevent STI operation. If the STI is initiated while in the REM Run mode by loading the status registers, the interrupt starts timing from the end of the program scan in which the status registers were loaded. |
| S31 to S32 | Reserved | NA | NA |

^① Address is not shown in HHP data monitor.

Function Codes

The table below provides the function codes for each instruction. The instructions are listed in alphabetical order.

| Mnemonic | HHP Display | Function Code | Name | Instruction Type | |
|-------------------|-------------|---------------|--------------------------------------|-----------------------|------------|
| ADD | ① | 80 | Add | Math | |
| ANB | ① | 13 | And Block | Basic | |
| AND (bit input) | →← | 22 | And | Basic | |
| AND (word output) | ① | 108 | And | Data Handling | |
| ANI | →← | 23 | And Inverted | Basic | |
| BSL | ① | 150 | Bit Shift Left | Application Specific | |
| BSR | ① | 151 | Bit Shift Right | Application Specific | |
| CLR | ① | 85 | Clear | Math | |
| COP | ① | 104 | File Copy | Data Handling | |
| CTD | ① | 6 | Count Down | Basic | |
| CTU | ① | 5 | Count Up | Basic | |
| DCD | ① | 102 | Decode 4 to 1 of 16 | Data Handling | |
| DDV | ① | 84 | Double Divide | Math | |
| DIV | ① | 83 | Divide | Math | |
| ENC | ① | 103 | Encode 1 of 16 to 4 | Data Handling | |
| EQU | AND EQU | →EQU← | 51 | Equal | Comparison |
| | LD EQU | ←EQU← | 50 | | |
| | OR EQU | ↳EQU↳ | 52 | | |
| FFL | ① | 113 | FIFO Load | Data Handling | |
| FFU | ① | 114 | FIFO Unload | Data Handling | |
| FLL | ① | 105 | Fill File | Data Handling | |
| FRD | ① | 101 | Convert from BCD | Data Handling | |
| GEQ | AND GEQ | →GEQ← | 66 | Greater Than or Equal | Comparison |
| | LD GEQ | ←GEQ← | 65 | | |
| | OR GEQ | ↳GEQ↳ | 67 | | |
| GRT | AND GRT | →GRT← | 63 | Greater Than | Comparison |
| | LD GRT | ←GRT← | 62 | | |
| | OR GRT | ↳GRT↳ | 64 | | |
| HSC | ① | 170 | High-Speed Counter | High-Speed Counter | |
| HSD | ① | 174 | High-Speed Counter Interrupt Disable | High-Speed Counter | |
| HSE | ① | 173 | High-Speed Counter Interrupt Enable | High-Speed Counter | |

① Multiple displays.

| Mnemonic | HHP Display | Function Code | Name | Instruction Type | |
|----------|----------------------------|----------------------------|----------------------------|-----------------------------|----------------------|
| HSL | ① | 171 | High-Speed Counter Load | High-Speed Counter | |
| IIM | ① | 138 | Immediate Input with Mask | Program Flow Control | |
| INT | LD INT | $\vdash \text{INT} \vdash$ | 158 | Interrupt Subroutine | Application Specific |
| IOM | ① | 139 | Immediate Output with Mask | Program Flow Control | |
| JMP | ① | 130 | Jump to Label | Program Flow Control | |
| JSR | ① | 132 | Jump to Subroutine | Program Flow Control | |
| LBL | LD LBL | $\vdash \text{LBL} \vdash$ | 131 | Label | Program Flow Control |
| LD | $\vdash \vdash$ | 20 | Load | Basic | |
| LDI | $\vdash \neg \vdash$ | 21 | Load Inverted | Basic | |
| LDT | $\vdash \text{LDT} \vdash$ | 26 | Load True | Basic | |
| LEQ | AND LEQ | $\neg \text{LEQ} \vdash$ | 60 | Less Than or Equal | Comparison |
| | LD LEQ | $\vdash \text{LEQ} \vdash$ | 59 | | |
| | OR LEQ | $\vdash \text{LEQ} \vdash$ | 61 | | |
| LES | AND LES | $\neg \text{LES} \vdash$ | 57 | Less Than | Comparison |
| | LD LES | $\vdash \text{LES} \vdash$ | 56 | | |
| | OR LES | $\vdash \text{LES} \vdash$ | 58 | | |
| LFL | ① | 115 | LIFO Load | Data Handling | |
| LFU | ① | 116 | LIFO Unload | Data Handling | |
| LIM | AND LIM | $\neg \text{LIM} \vdash$ | 72 | Limit Test | Comparison |
| | LD LIM | $\vdash \text{LIM} \vdash$ | 71 | | |
| | OR LIM | $\vdash \text{LIM} \vdash$ | 73 | | |
| MCR | ① | 135 | Master Control Reset | Program Flow Control | |
| MEQ | AND MEQ | $\neg \text{MEQ} \vdash$ | 69 | Masked Comparison for Equal | Comparison |
| | LD MEQ | $\vdash \text{MEQ} \vdash$ | 68 | | |
| | OR MEQ | $\vdash \text{MEQ} \vdash$ | 70 | | |
| MOV | ① | 106 | Move | Data Handling | |
| MPP | ① | 12 | Memory Pop | Basic | |
| MPS | ① | 10 | Memory Push | Basic | |
| MRD | ① | 11 | Memory Read | Basic | |
| MSG | ① | 200 | Message | Communication | |
| MUL | ① | 82 | Multiply | Math | |
| MVM | ① | 107 | Masked Move | Data Handling | |

① Multiple displays.

| Mnemonic | HHP Display | Function Code | Name | Instruction Type |
|--------------------------|-------------|---------------|---------------------------------------|----------------------|
| NEG | ① | 112 | Negate | Data Handling |
| AND NEQ | ┌─ NEQ ─┐ | 54 | Not Equal | Comparison |
| LD NEQ | ┌─ NEQ ─┐ | 53 | | |
| OR NEQ | └─ NEQ ─┘ | 55 | | |
| NOT | ① | 111 | Not | Data Handling |
| OR (bit input) | └─┘ | 24 | Or | Basic |
| OR (word output) | ① | 109 | Or | Data Handling |
| ORB | ① | 14 | Or Block | Basic |
| ORI | └─┘ | 25 | Or Inverted | Basic |
| ORT | └─ ORT ─┘ | 27 | Or True | Basic |
| AND OSR | ┌─ OSR ─┐ | 29 | One-Shot Rising | Basic |
| LD OSR | ┌─ OSR ─┐ | 28 | | |
| OUT (basic instruction) | ─()─ | 40 | Output | Basic |
| OUT (high-speed counter) | ① | 40 | Update High-Speed Counter Accumulator | High-Speed Counter |
| RAC | ① | 172 | High-Speed Counter Reset Accumulator | High-Speed Counter |
| RES (timer/counter) | ① | 7 | Reset | Basic |
| RES (high-speed counter) | ① | 7 | High-Speed Counter Reset | High-Speed Counter |
| RET | ① | 134 | Return from Subroutine | Program Flow Control |
| RST | ─(U)─ | 42 | Reset | Basic |
| RTO | ① | 2 | Retentive Timer | Basic |
| SBR LD SBR | ┌─ SBR ─┐ | 133 | Subroutine | Program Flow Control |
| SCL | ① | 87 | Scale Data | Math |
| SET | ─(L)─ | 41 | Set | Basic |
| SQC | ① | 153 | Sequencer Compare | Application Specific |
| SQL | ① | 154 | Sequencer Load | Application Specific |
| SQO | ① | 152 | Sequencer Output | Application Specific |
| SQR | ① | 86 | Square Root | Math |
| STD | ① | 155 | Selectable Timer Interrupt Disable | Application Specific |
| STE | ① | 156 | Selectable Timer Interrupt Enable | Application Specific |
| STS | ① | 157 | Selectable Timer Interrupt Start | Application Specific |

① Multiple displays.

| Mnemonic | HHP Display | Function Code | Name | Instruction Type |
|----------|-------------|---------------|-----------------|----------------------|
| SUB | ① | 81 | Subtract | Math |
| SUS | ① | 137 | Suspend | Program Flow Control |
| TND | ① | 136 | Temporary End | Program Flow Control |
| TOD | ① | 100 | Convert to BCD | Data Handling |
| TOF | ① | 1 | Timer Off-Delay | Basic |
| TON | ① | 0 | Timer On-Delay | Basic |
| XOR | ① | 110 | Exclusive Or | Data Handling |

① Multiple displays.

Instruction Execution Times and Memory Usage

The table below lists the execution times and memory usage for the controller instructions. Any instruction that takes longer than 15 μ s (true or false execution time) to execute performs a poll for user interrupts.

| Mnemonic | False Execution Time (approx. μ seconds) | True Execution Time (approx. μ seconds) | Memory Usage (user words) | Name | Instruction Type |
|-------------------|--|---|---------------------------|---------------------|----------------------|
| ADD | 6.78 | 33.09 | 1.50 | Add | Math |
| ANB | 0.40 | | 0.25 | And Block | Basic |
| AND (bit input) | 2.12 | 1.94 | 1.00 | And | Basic |
| AND (word output) | 6.78 | 34.00 | 1.50 | And | Data Handling |
| ANI | 2.12 | 1.94 | 1.00 | And Inverted | Basic |
| BSL | 19.80 | $53.71 + 5.24 \times \text{position value}$ | 2.00 | Bit Shift Left | Application Specific |
| BSR | 19.80 | $53.34 + 3.98 \times \text{position value}$ | 2.00 | Bit Shift Right | Application Specific |
| CLR | 4.25 | 20.80 | 1.00 | Clear | Math |
| COP | 6.60 | $27.31 + 5.06/\text{word}$ | 1.50 | File Copy | Data Handling |
| CTD | 27.22 | 32.19 | 1.00 | Count Down | Basic |
| CTU | 26.67 | 29.84 | 1.00 | Count Up | Basic |
| DCD | 6.78 | 27.67 | 1.50 | Decode 4 to 1 of 16 | Data Handling |
| DDV | 6.78 | 157.06 | 1.00 | Double Divide | Math |
| DIV | 6.78 | 147.87 | 1.50 | Divide | Math |
| ENC | 6.78 | 54.80 | 1.50 | Encode 1 of 16 to 4 | Data Handling |
| EQU | AND EQU | 7.00 | 21.92 | Equal | Comparison |
| | LD EQU | 6.60 | 21.52 | | |
| | OR EQU | 7.00 | 21.92 | | |

| Mnemonic | False Execution Time (approx. μ seconds) | True Execution Time (approx. μ seconds) | Memory Usage (user words) | Name | Instruction Type | |
|----------|---|--|------------------------------|---|----------------------|----------------------|
| FFL | 33.67 | 61.13 | 1.50 | FIFO Load | Data Handling | |
| FFU | 34.90 | $73.78 + 4.34 \times \text{position value}$ | 1.50 | FIFO Unload | Data Handling | |
| FLL | 6.60 | $26.86 + 3.62/\text{word}$ | 1.50 | Fill File | Data Handling | |
| FRD | 5.52 | 56.88 | 1.00 | Convert from BCD | Data Handling | |
| GEQ | AND GEQ | 7.00 | 24.00 | Greater Than or Equal | Comparison | |
| | LD GEQ | 6.60 | 23.60 | | | |
| | OR GEQ | 7.00 | 24.00 | | | |
| GRT | AND GRT | 7.00 | 24.00 | Greater Than | Comparison | |
| | LD GRT | 6.60 | 23.60 | | | |
| | OR GRT | 7.00 | 24.00 | | | |
| HSC | 21.00 | 21.00 | 1.00 | High-Speed Counter | High-Speed Counter | |
| HSD | 7.00 | 8.00 | 1.25 | High-Speed Counter Interrupt Disable | High-Speed Counter | |
| HSE | 7.00 | 10.00 | 1.25 | High-Speed Counter Interrupt. Enable | High-Speed Counter | |
| HSL | 7.00 | 66.00 | 1.50 | High-Speed Counter Load | High-Speed Counter | |
| IIM | 6.78 | 35.72 | 1.50 | Immediate Input with Mask | Program Flow Control | |
| INT | LD INT | 0.99 | 1.45 | 0.50 | Interrupt Subroutine | Application Specific |
| IOM | 6.78 | 41.59 | 1.50 | Immediate Output with Mask | Program Flow Control | |
| JMP | 6.78 | 9.04 | 1.00 | Jump to Label | Program Flow Control | |
| JSR | 4.25 | 22.24 | 1.00 | Jump to Subroutine | Program Flow Control | |
| LBL | LD LBL | 0.99 | 1.45 | 0.50 | Label | Program Flow Control |
| LD | 1.72 | 1.54 | 0.75 | Load | Basic | |
| LDI | 1.72 | 1.54 | 0.75 | Load Inverted | Basic | |
| LDT | n/a | 1.54 | 0.75 | Load True | Basic | |
| LEQ | AND LEQ | 7.00 | 24.00 | Less Than or Equal | Comparison | |
| | LD LEQ | 6.60 | 23.60 | | | |
| | OR LEQ | 7.00 | 24.00 | | | |
| LES | AND LES | 7.00 | 24.00 | Less Than | Comparison | |
| | LD LES | 6.60 | 23.60 | | | |
| | OR LES | 7.00 | 24.00 | | | |
| LFL | 33.67 | 61.13 | 1.50 | LIFO Load | Data Handling | |
| LFU | 35.08 | 64.20 | 1.50 | LIFO Unload | Data Handling | |

Appendix B
Programming Reference

| Mnemonic | False Execution Time (approx. μ seconds) | True Execution Time (approx. μ seconds) | Memory Usage (user words) | Name | Instruction Type |
|--------------------------|---|--|------------------------------|---|----------------------|
| LIM | AND LIM | 8.09 | 37.33 | Limit Test | Comparison |
| | LD LIM | 7.69 | 36.93 | | |
| | OR LIM | 8.09 | 37.33 | | |
| MCR | 4.07 | 3.98 | 0.50 | Master Control Reset | Program Flow Control |
| MEQ | AND MEQ | 8.09 | 28.79 | Masked Comparison for Equal | Comparison |
| | LD MEQ | 7.69 | 28.39 | | |
| | OR MEQ | 8.09 | 28.79 | | |
| MOV | 6.78 | 25.05 | 1.50 | Move | Data Handling |
| MPP | 0.40 | | 0.25 | Memory Pop | Basic |
| MPS | 0.40 | | 0.25 | Memory Push | Basic |
| MRD | 0.40 | | 0.25 | Memory Read | Basic |
| MSG | 26 | 180 ^{①②} | 34.75 | Message | Communication |
| MUL | 6.78 | 57.96 | 1.50 | Multiply | Math |
| MVM | 6.78 | 33.28 | 1.50 | Masked Move | Data Handling |
| NEG | 6.78 | 29.48 | 1.50 | Negate | Data Handling |
| NEQ | AND NEQ | 7.00 | 21.92 | Not Equal | Comparison |
| | LD NEQ | 6.60 | 21.52 | | |
| | OR NEQ | 7.00 | 21.92 | | |
| NOT | 6.78 | 28.21 | 1.00 | Not | Data Handling |
| OR (bit input) | 2.12 | 1.94 | 1.00 | Or | Basic |
| OR (word output) | 6.78 | 33.68 | 1.50 | Or | Data Handling |
| ORB | 0.40 | | 0.25 | Or Block | Basic |
| ORI | 2.12 | 1.94 | 1.00 | Or Inverted | Basic |
| ORT | n/a | 1.94 | 1.00 | Or True | Basic |
| OSR | AND OSR | 11.88 | 13.42 | One-Shot Rising | Basic |
| | LD OSR | 11.48 | 13.02 | | |
| OUT (basic instruction) | 4.43 | 4.43 | 0.75 | Output | Basic |
| OUT (high-speed counter) | 7.00 | 12.00 | 0.75 | Update High-Speed Counter Image Accumulator | High-Speed Counter |
| RAC | 6.00 | 56.00 | 1.00 | High-Speed Counter Reset Accumulator | High-Speed Counter |
| RES (timer/counter) | 4.25 | 15.19 | 1.00 | Reset | Basic |
| RES (high-speed counter) | 6.00 | 51.00 | 1.00 | High-Speed Counter Reset | High-Speed Counter |

① This only includes the amount of time needed to set up the operation requested. It does not include the time it takes to service the actual communication, as this time varies with each network configuration. As an example, 144ms is the actual communication service time for the following configuration: 3 nodes on DH-485 (2=MicroLogix 1000 programmable controllers and 1=PLC-500 A.I. Series™ programming software), running at 19.2K baud, with 2 words per transfer.

② Add 7.3 μ seconds per word for MSG instructions that perform writes.

| Mnemonic | False Execution Time (approx. μ seconds) | True Execution Time (approx. μ seconds) | Memory Usage (user words) | Name | Instruction Type |
|-----------------|---|--|------------------------------|---------------------------------------|----------------------|
| RET | 3.16 | 31.11 | 0.50 | Return from Subroutine | Program Flow Control |
| RST | 3.16 | 4.97 | 0.75 | Reset | Basic |
| RTO | 27.49 | 38.34 | 1.00 | Retentive Timer | Basic |
| SBR LD SBR | 0.99 | 1.45 | 0.50 | Subroutine | Program Flow Control |
| SCL | 6.78 | 169.18 | 1.75 | Scale Data | Math |
| SET | 3.16 | 4.97 | 0.75 | Set | Basic |
| SQC | 27.40 | 60.52 | 2.00 | Sequencer Compare | Application Specific |
| SQL | 28.12 | 53.41 | 2.00 | Sequencer Load | Application Specific |
| SOO | 27.40 | 60.52 | 2.00 | Sequencer Output | Application Specific |
| SQR | 6.78 | 71.25 | 1.25 | Square Root | Math |
| STD | 3.16 | 6.69 | 0.50 | Selectable Timer Interrupt Disable | Application Specific |
| STE | 3.16 | 10.13 | 0.50 | Selectable Timer Interrupt Enable | Application Specific |
| STS | 6.78 | 24.59 | 1.25 | Selectable Timer Interrupt Start | Application Specific |
| SUB | 6.78 | 33.52 | 1.50 | Subtract | Math |
| SUS | 7.87 | 10.85 | 1.50 | Suspend | Program Flow Control |
| TND | 3.16 | 7.78 | 0.50 | Temporary End | Program Flow Control |
| TOD | 6.78 | 49.64 | 1.00 | Convert to BCD | Data Handling |
| TOF | 31.65 | 39.42 | 1.00 | Timer Off-Delay | Basic |
| TON | 30.38 | 38.34 | 1.00 | Timer On-Delay | Basic |
| XOR | 6.92 | 33.64 | 1.50 | Exclusive Or | Data Handling |

User Interrupt Latency

The user interrupt latency is the maximum time from when an interrupt condition occurs (e.g., STI expires or HSC preset is reached) to when the user interrupt subroutine begins executing (assumes that there are no other interrupt conditions present). The following table lists interrupt conditions and the corresponding time to execute the interrupt subroutine:

| User Interrupt Condition | Time to Execute |
|----------------------------|-----------------|
| HSC preset is reached | 183 μ s |
| STI expires | 74 μ s |
| Comms link layer interrupt | 644 μ s |
| System overhead interrupt | 20 μ s |

To calculate the user interrupt latency when you are communicating with the controller, add 664 μ s to one or both of the following values:

- 183 μ s
- 74 μ s

For example, if you are communicating with the controller, the worst case interrupt latency is 921 μ s (644 μ s + 257 μ s).

To calculate the user interrupt latency when you are *not* communicating with the controller, add 20 μ s to one or both of the following values:

- 183 μ s
- 74 μ s

For example, if you are *not* communicating with the controller, the worst case interrupt latency is 277 μ s (20 μ s + 257 μ s).

Estimating Memory Usage for Your Control System

Use the following to calculate memory usage for your control system.

_____ 177

_____ 110

Total Memory Usage: _____

1. Determine the total instruction words used by the instructions in your program and enter the result. Refer to the table on page B–16.
2. Multiply the total number of rungs by 0.75 and enter the result. Do not count the Start of File or End of File screens in each file.
3. To account for controller overhead, use 177.
4. To account for application data, use 110.
5. Total steps 1 through 4. This is the estimated total memory usage of your application system. Remember, this is an estimate, actual compiled programs may differ by $\pm 12\%$.
6. To determine the estimated amount of memory remaining in the controller you have selected, do the following:
Subtract the total memory usage from 1024.

_____ 1024

Total Memory Usage
(from above): _____

Total Memory
Remaining: _____

The result of this calculation is the estimated total memory remaining in your selected controller.

Important: The calculated memory usage may vary from the actual compiled program by $\pm 12\%$.

Execution Time Worksheet

Use this worksheet to calculate your execution time for logic program.

| Procedure | Maximum Scan Time |
|---|--|
| 1. Input scan time, output scan time, housekeeping time, and forcing. | 210 μs (discrete) 330 μs with forcing (analog) <u>250</u> μs without forcing (analog) |
| 2. Estimate your program scan time: A. Count the number of program rungs in your logic program and multiply by 6. B. Add up your program execution times when all instructions are true. Include interrupt routines in this calculation. ^① | _____ _____ μs |
| 3. Estimate your controller scan time: A. Without communications, add sections 1 and 2 B. With communications, add sections 1 and 2 and multiply by 1.05 | _____ μs _____ μs |
| 4. To determine your maximum scan time in ms, divide your controller scan time by 1000. | _____ ms |

^① If a subroutine executes more than once per scan, include each subroutine execution scan time.

Valid Addressing Modes and File Types for Instruction Parameters

This appendix lists all of the available programming instructions along with their parameters, valid addressing modes, and file types.

Available File Types

The following file types are available:

- O Output
- I Input
- S Status
- B Binary
- T Timer
- C Counter
- R Control
- N Integer

All file types are word addresses, unless otherwise specified.

Available Addressing Modes

The following addressing modes are available:

- immediate
- direct
- indexed direct

Immediate Addressing

Indicates that a constant is a valid file type.

Direct Addressing

The data stored in the specified address is used in the instruction. For example:

N7:0 T4:8.ACC
ST20:5

Indexed Direct Addressing

You may specify an address as being indexed by placing the “#” character in front of the address. When an address of this form is encountered in the program, the processor takes the element number of the address and adds to it the value contained in the Index Register S:24, then uses the result as the actual address. For example:

#N7:10 where S:24 = 15
The actual address used by the instruction is N7:25.

Appendix C

Valid Addressing Modes and File Types for Instruction Parameters

| Instruction | Description | Instruction Parameters | Valid Addressing Mode(s) | Valid File Types | Valid Value Ranges |
|-------------|-------------------|------------------------|-------------------------------------|---------------------------------------|---|
| ADD | Add | source A | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| AND | And (bit input) | source bit | direct | O, I, S, B, T, C, R, N (bit level) | Not Applicable |
| AND | And (word output) | source A | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| ANI | And Inverted | source bit | direct | O, I, S, B, T, C, R, N (bit level) | Not Applicable |
| BSL | Bit Shift Left | file | indexed direct | O, I, S, B, N | Not Applicable |
| | | control | direct | R (element level) | Not Applicable |
| | | bit address | direct | O, I, S, B, T, C, R, N (bit level) | Not Applicable |
| | | length | (contained in the control register) | | 0-2048 |
| BSR | Bit Shift Right | file | indexed direct | O, I, S, B, N | Not Applicable |
| | | control | direct | R (element level) | Not Applicable |
| | | bit address | direct | O, I, S, B, T, C, R, N (bit level) | Not Applicable |
| | | length | (contained in the control register) | | 0-2048 |
| CLR | Clear | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| COP | Copy File | source | indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | destination | indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | length | immediate | | 1-128; 1-42 when destination is T,C,R |
| CTD | Count Down | counter | direct | C (element level) | Not Applicable |
| | | preset | (contained in the counter register) | | -32,768-32,767 |
| | | accum | (contained in the counter register) | | -32,768-32,767 |

Appendix C

Valid Addressing Modes and File Types for Instruction Parameters

| Instruction | Description | Instruction Parameters | Valid Addressing Mode(s) | Valid File Types | Valid Value Ranges |
|-------------|---------------------|------------------------|-------------------------------------|------------------------|-------------------------------|
| CTU | Count Up | counter | direct | C (element level) | Not Applicable |
| | | preset | (contained in the counter register) | | –32,768–32,767 |
| | | accum | (contained in the counter register) | | –32,768–32,767 |
| DCD | Decode 4 to 1 of 16 | source | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| DDV | Double Divide | source | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| DIV | Divide | source A | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 f-min–f-max |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 f-min–f-max |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| ENC | Encode 1 of 16 to 4 | source | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| EQU | Equal | source A | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 f-min–f-max |
| FFL | FIFO Load | source | direct, indexed direct ^① | O, I, S, B, T, C, R, N | –32,768–32,767 |
| | | FIFO array | indexed direct | O, I, S, B, N | Not Applicable |
| | | FIFO control | direct | R (element level) | Not Applicable |
| | | length | (contained in the control register) | | 1–128 |
| | | position | (contained in the control register) | | 0–127 |
| FFU | FIFO Unload | FIFO array | indexed direct | O, I, S, B, N | Not Applicable |
| | | destination | direct, indexed direct ^① | O, I, S, B, T, C, R, N | Not Applicable |
| | | FIFO control | direct | R (element level) | Not Applicable |
| | | length | (contained in the control register) | | 1–128 |
| | | position | (contained in the control register) | | 0–127 |

^① Indexed addressing is not allowed when using T, C, or R addresses.

Appendix C

Valid Addressing Modes and File Types for Instruction Parameters

| Instruction | Description | Instruction Parameters | Valid Addressing Mode(s) | Valid File Types | Valid Value Ranges |
|-------------|------------------------------|------------------------|--|---|---|
| FLL | Fill File | source | direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| | | destination | indexed direct | O, I, S, B, T, C, R, N (element level) | Not Applicable |
| | | length | immediate | | 1-128; 1-42 when destination is T,C,R |
| FRD | Convert from BCD | source | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| GEO | Greater Than or Equal | source A | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| GRT | Greater Than | source A | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| HSC | High-Speed Counter | type | immediate | | 0-7, where: 0=up 1=up&reset/hold 2=pulse/direction 3=pule/direction & reset/hold 4=up/down 5=up/down & reset/hold 6=encoder 7=encoder & reset/hold |
| | | counter | direct | C5:0. C5:1 (element level) | Not Applicable |
| | | preset | (contained in the counter register) | | -32,768-32,767 |
| | | accum | (contained in the counter register) | | -32,768-32,767 |
| HSD | HSC Interrupt Disable | counter | direct | C | Not Applicable |
| HSE | HSC Interrupt Enable | counter | direct | C | Not Applicable |
| HSL | HSC Load | counter | direct | C | Not Applicable |
| | | source | direct | B, N | Not Applicable |
| | | length | | | always 5 |
| IIM | Immediate Input with Mask | slot | direct | I | Not Applicable |
| | | mask | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | length | immediate | | 1-10 |

Appendix C

Valid Addressing Modes and File Types for Instruction Parameters

| Instruction | Description | Instruction Parameters | Valid Addressing Mode(s) | Valid File Types | Valid Value Ranges |
|-------------|----------------------------|------------------------|--|-------------------------------------|-------------------------------|
| INT | Interrupt Subroutine | | | | Not Applicable |
| IOM | Immediate Output with Mask | slot | direct | O | Not Applicable |
| | | mask | direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | length | immediate | | 1-32 |
| JMP | Jump | label number | immediate | | 0-999 |
| JSR | Jump to Subroutine | subroutine file number | immediate | | 3-255 |
| LBL | Label | label number | immediate | | 0-999 |
| LD | Load | source bit | direct | O, I, S, B, T, C, R, N (bit level) | Not Applicable |
| LDI | Load Inverted | source bit | direct | O, I, S, B, T, C, R, N (bit level) | Not Applicable |
| LEQ | Less Than or Equal To | source A | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| LES | Less Than | source A | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| LFL | LIFO Load | source | immediate, direct, indexed direct ^① | O, I, S, B, T, C, R, N ^① | -32,768-32,767 |
| | | LIFO array | indexed direct | O, I, S, B, N | Not Applicable |
| | | LIFO control | direct | R (element level) | Not Applicable |
| | | length | (contained in the control register) | | 1-128 |
| | | position | (contained in the control register) | | 0-127 |
| LFU | LIFO Unload | LIFO array | indexed direct | O, I, S, B, N | Not Applicable |
| | | destination | direct, indexed direct ^① | O, I, S, B, T, C, R, N | Not Applicable |
| | | LIFO control | direct | R (element level) | Not Applicable |
| | | length | (contained in the control register) | | 1-128 |
| | | position | (contained in the control register) | | 0-127 |

^① Indexed addressing is not allowed when using T, C, or R addresses.

Appendix C

Valid Addressing Modes and File Types for Instruction Parameters

| Instruction | Description | Instruction Parameter | Valid Addressing Mode(s) | Valid File Types | Valid Value Ranges |
|-------------|---------------------------|-----------------------|-------------------------------------|--------------------------|-----------------------------------|
| LIM | Limit Test | low limit | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| | | test | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| | | high limit | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| MCR | Master Control Reset | | | | Not Applicable |
| MEQ | Mask Comparison for Equal | source | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | source mask | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | compare | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 |
| MOV | Move | source | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| MSG | Message | read/write | immediate | | 0=read,1=write |
| | | target device | immediate | | 2=500CPU, 4=485CIF |
| | | control block | direct | N | Not Applicable |
| | | control block length | immediate | | 7 |
| | | local address | direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | target node | (contained in the control register) | | 0-254 for DF1; 1-31 for DH-485 |
| | | target address | direct | O, I, S, B, T, C, R, N | 0-255 |
| | | message length | | T, C, R I, O, S, B, N | 1-13 1-41 |
| MUL | Multiply | source A | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| MVM | Masked Move | source | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | source mask | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| NEG | Negate | source | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |

Appendix C

Valid Addressing Modes and File Types for Instruction Parameters

| Instruction | Description | Instruction Parameter | Valid Addressing Mode(s) | Valid File Types | Valid Value Ranges |
|-------------|-----------------------------|-----------------------|-----------------------------------|---------------------------------------|-------------------------------|
| NEQ | Not Equal | source A | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 f-min–f-max |
| NOT | Logical NOT | source | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| OR | Or (bit input) | source bit | direct | O, I, S, B, T, C, R, N (bit level) | Not Applicable |
| OR | Or (word output) | source A | direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 |
| | | source B | direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| ORI | Or Inverted | source bit | direct | O, I, S, B, T, C, R, N (bit level) | Not Applicable |
| OSR | One-Shot Rising | bit address | direct | O, I, S, B, T, C, R, N | Not Applicable |
| OUT | Output (or Output Energize) | bit address | direct | O, I, S, B, T, C, R, N | Not Applicable |
| RAC | HSC Reset Accumulator | counter | direct | C | Not Applicable |
| | | source | immediate, direct | O, I, S, B, T, C, R, N | –32,768–32,767 f-min–f-max |
| RES | Reset (timer/counter) | structure | direct | T, C, R (element level) | Not Applicable |
| RES | Reset (high-speed counter) | structure | direct | T, C, R (element level) | Not Applicable |
| RET | Return | | | | Not Applicable |
| RST | Reset (or Output Unlatch) | bit address | direct | O, I, S, B, T, C, R, N | Not Applicable |
| RTO | Retentive Timer | timer | direct | T (element level) | Not Applicable |
| | | time base | immediate | | 0.01 or 1.00 |
| | | preset | (contained in the timer register) | | 0–32,767 |
| | | accum | (contained in the timer register) | | 0–32,767 |
| SBR | Subroutine | | | | Not Applicable |

Appendix C

Valid Addressing Modes and File Types for Instruction Parameters

| Instruction | Description | Instruction Parameter | Valid Addressing Mode(s) | Valid File Types | Valid Value Ranges |
|-------------|--------------------------|-----------------------|--|------------------------|-------------------------------|
| SCL | Scale Data | source | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| | | rate | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | offset | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| SET | Set (or Output Latch) | bit address | direct | O, I, S, B, T, C, R, N | Not Applicable |
| SOC | Sequencer Compare | file | indexed direct | O, I, S, B, N | Not Applicable |
| | | mask | immediate, direct, indexed direct ^① | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | source | direct, indexed direct ^① | O, I, S, B, T, C, R, N | Not Applicable |
| | | control | direct | R (element level) | Not Applicable |
| | | length | (contained in the control register) | | 1-255 |
| | | position | (contained in the control register) | | 0-255 |
| SQL | Sequencer Load | file | indexed direct | O, I, S, B, N | Not Applicable |
| | | source | direct, indexed direct ^① | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | control | direct | R (element level) | Not Applicable |
| | | length | (contained in the control register) | | 1-255 |
| | | position | (contained in the control register) | | 0-255 |
| SQO | Sequencer Output | file | indexed direct | O, I, S, B, N | Not Applicable |
| | | mask | direct, indexed direct ^① | O, I, S, B, T, C, R, N | -32,768-32,767 |
| | | destination | direct, indexed direct ^① | O, I, S, B, T, C, R, N | Not Applicable |
| | | control | direct | R (element level) | Not Applicable |
| | | length | | | 1-255 |
| | | position | | | 0-255 |
| SQR | Square Root | source | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | -32,768-32,767 f-min-f-max |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| STD | Selectable Timed Disable | | | | Not Applicable |
| STE | Selectable Timed Enable | | | | Not Applicable |

^① Indexed addressing is not allowed when using T, C, or R addresses.

Appendix C

Valid Addressing Modes and File Types for Instruction Parameters

| Instruction | Description | Instruction Parameter | Valid Addressing Mode(s) | Valid File Types | Valid Value Ranges |
|-------------|------------------------|-----------------------|-----------------------------------|------------------------|-------------------------------|
| STS | Selectable Timed Start | file | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | always equal 5 |
| | | time | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | 0–255 |
| SUB | Subtract | source A | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 f-min–f-max |
| | | source B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 f-min–f-max |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |
| SUS | Suspend | suspend ID | immediate, | | –32,768–32,767 |
| TND | Temporary End | | | | Not Applicable |
| TOD | Convert to BCD | source | direct, indexed direct | O, I, S, B, T, C, R, N | |
| | | destination | direct | O, I, S, B, T, C, R, N | Not Applicable |
| TOF | Timer Off-Delay | timer | direct | T (element level) | Not Applicable |
| | | time base | immediate | | 0.01 or 1.00 |
| | | preset | (contained in the timer register) | | 0–32,767 |
| | | accum | (contained in the timer register) | | 0–32,767 |
| TON | Timer On-Delay | timer | direct | T (element level) | Not Applicable |
| | | time base | immediate | | 0.01 or 1.00 |
| | | preset | (contained in the timer register) | | 0–32,767 |
| | | accum | (contained in the timer register) | | 0–32,767 |
| XOR | Exclusive OR | address A | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 |
| | | address B | immediate, direct, indexed direct | O, I, S, B, T, C, R, N | –32,768–32,767 |
| | | destination | direct, indexed direct | O, I, S, B, T, C, R, N | Not Applicable |

Understanding the Communication Protocols

Use the information in this appendix to understand the differences in communication protocols. The following protocols are supported from the RS-232 communication channel:

- **DF1 Full-Duplex and DF1 Half-Duplex Slave**
All MicroLogix 1000 controllers support the DF1 full-duplex protocol. Series D or later discrete and all MicroLogix 1000 analog controllers also support DF1 half-duplex slave protocol. Note that the MicroLogix 1000 HHP cannot be used to select or configure the DF1 half-duplex slave protocol.
- **DH-485**
Series C or later discrete and all MicroLogix 1000 analog controllers can communicate on DH-485 networks using an AIC+ Advanced Interface Converter.

For information about required network connecting equipment, see chapter 3, *Connecting the System*.

RS-232 Communication Interface

RS-232 is an Electronics Industries Association (EIA) standard that specifies the electrical, mechanical, and functional characteristics for serial binary communication. It provides you with a variety of system configuration possibilities. (RS-232 is a definition of electrical characteristics; it is *not* a protocol.)

One of the biggest benefits of the RS-232 interface is that it lets you integrate telephone and radio modems into your control system (using the appropriate DF1 protocol only; not DH-485 protocol). The distance over which you are able to communicate with certain system devices is virtually limitless.

DF1 Full-Duplex Protocol

DF1 Full-Duplex communication protocol combines data transparency (ANSI — American National Standards Institute — specification subcategory D1) and two-way simultaneous transmission with embedded responses (subcategory F1).

The MicroLogix 1000 controllers support the DF1 Full-Duplex protocol via RS-232 connection to external devices, such as computers, the Hand-Held Programmer (catalog number 1761-HHP-B30), or other MicroLogix 1000 controllers. (For information on connecting to the Hand-Held Programmer, see chapter 3, *Connecting the System*).

DF1 Full-Duplex Operation

DF1 Full-Duplex protocol (also referred to as DF1 point-to-point protocol) is useful where RS-232 point-to-point communication is required. This type of protocol supports simultaneous transmissions between two devices in both directions. DF1 protocol controls message flow, detects and signals errors, and retries if errors are detected.

DF1 Full-Duplex Configuration Parameters

When the system mode driver is DF1 Full-Duplex, the following parameters can be changed:

| Parameter | Options | Default |
|----------------------------|--|------------------------|
| Baud Rate | Toggles between the communication rate of 300, 600, 1200, 2400, 4800 ^① , 9600, 19200, and 38400 ^① . | 9600 ^② |
| Node Address | Valid range is 0–254 decimal for MicroLogix 1000 Series C and later discrete and all MicroLogix 1000 analog. Not configurable for MicroLogix 1000 Series A and B discrete. | 1 |
| Parity | None | No Parity |
| Stop Bits | None | 1 |
| Error Detection | None | CRC |
| DLE NAK received | None | N retries ^③ |
| DLE ENQ received | None | N retries ^③ |
| ACK Timeout | None | 1 s |
| Duplicate Packet Detection | None | Enabled |
| Control Line | None | No Handshaking |
| Embedded Responses | None | Enabled |

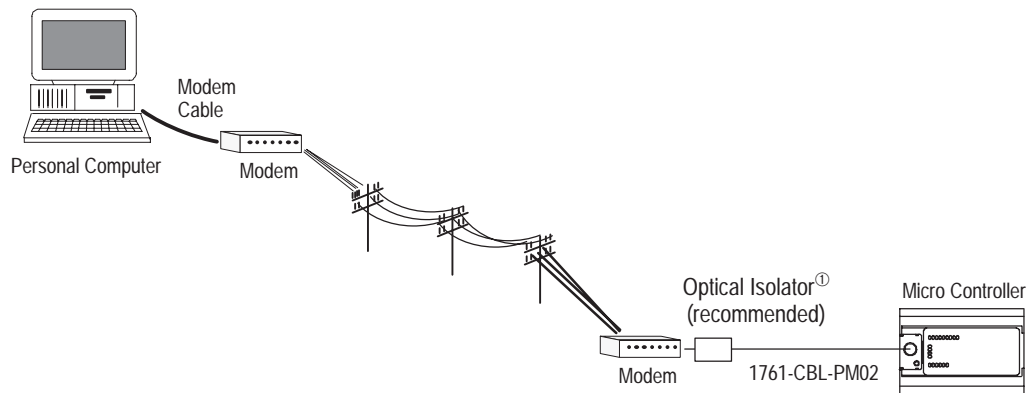
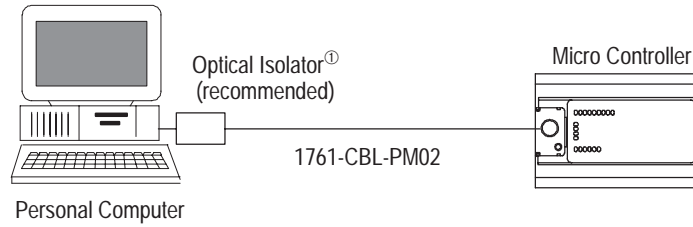
^① Applicable only to MicroLogix 1000 Series D or later discrete and all MicroLogix 1000 analog controllers.

^② If retentive communication data is lost, default is 1200 for MicroLogix 1000 Series A, B, or C discrete only. For MicroLogix 1000 Series D or later discrete and all MicroLogix 1000 analog, if retentive communication data is lost, baud rate defaults to 9600.

^③ N=255 for MicroLogix 1000 Series A and B discrete. N=6 for MicroLogix 1000 Series C and later discrete and all MicroLogix 1000 analog.

Example: DF1 Full-Duplex Connections

For information about required network connecting equipment, see chapter 3, Connecting the System.



① We recommend using an AIC+, catalog number 1761-NET-AIC, as your optical isolator. See page 3-8 and 3-9 for specific AIC+ cabling information.

DF1 Half-Duplex Slave Protocol

DF1 half-duplex slave protocol provides a multi-drop single master/multiple slave network. In contrast to DF1 full-duplex, communication takes place in one direction at a time. You can use the RS-232 port on the MicroLogix as both a half-duplex programming port, as well as a half-duplex peer-to-peer messaging port.

The master device initiates all communication by “polling” each slave device. The slave device may only transmit message packets when it is polled by the master. It is the master’s responsibility to poll each slave on a regular and sequential basis to allow slaves to send message packets back to the master. During a polling sequence, the master polls a slave either repeatedly until the slave indicates that it has no more message packets to transmit or just one time per polling sequence, depending on how the master is configured.

An additional feature of the DF1 half-duplex protocol is that it is possible for a slave device to enable a MSG instruction in its ladder program to send or request data to/from another slave. When the initiating slave is polled, the MSG instruction command packet is sent to the master. The master recognizes that the command packet is not intended for it but for another slave, so the master immediately rebroadcasts the command packet to the intended slave. When the intended slave is polled, it sends a reply packet to the master with the data the first slave requested. The master again recognizes that the reply packet is intended for another slave, so the master immediately rebroadcasts the reply packet to that slave. This slave-to-slave transfer is a function of the master device and is also used by programming software to upload and download programs to processors on the DF1 half-duplex link.

Several Allen-Bradley products support DF1 half-duplex master protocol. They include the SLC 5/03™, SLC 5/04™ and SLC 5/05™, and enhanced PLC-5® processors. Rockwell Software WINTelligent LINX™ and RSLinx (version 2.x and higher) also support DF1 half-duplex master protocol.

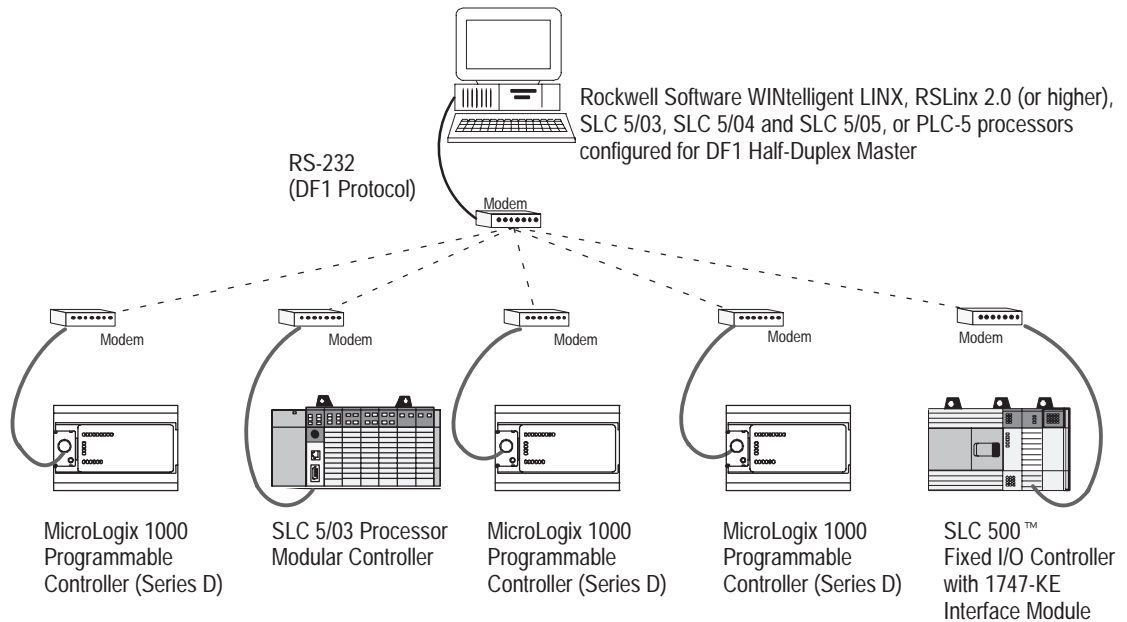
Typically, the master maintains an active node table that indicates which slaves are active (slaves that responded the last time they were polled) and which slaves are inactive (slaves that did not respond the last time they were polled). The active slaves are polled on a regular basis. The inactive slaves are only polled occasionally to check if any have come back online.

DF1 half-duplex supports up to 255 devices (address 0 to 254) with address 255 reserved for master broadcasts. The MicroLogix supports broadcast reception but cannot initiate a broadcast command. The MicroLogix supports half-duplex modems using Request-To-Send/Clear-To-Send (RTS/CTS) hardware handshaking.

DF1 Half-Duplex Slave Configuration Parameters

When the system mode driver is DF1 half-duplex slave the following parameters can be viewed and changed only when the programming software is online with the processor. The DF1 half-duplex slave parameters are not stored as part of the controller downloadable image (*with the exception of the baud rate and node address*). If a failed MicroLogix 1000 controller is replaced and the backed-up controller image is downloaded to the replacement controller, these parameters remain at default until manually changed. Therefore, be sure to fully document any non-default settings to the DF1 half-duplex slave configuration parameters.

| Parameter | Description | Default |
|----------------------------|---|----------------|
| Baud Rate | Toggles between the communication rate of 300, 600, 1200, 2400, 4800, 9600, 19,200, and 38.4K. | 9600 |
| Node Address | Valid range is 0–254 decimal. | 1 |
| Control Line | Toggles between No Handshaking and Half-Duplex Modem. | No Handshaking |
| Duplicate Packet Detection | Detects and eliminates duplicate responses to a message. Duplicate packets may be sent under “noisy” communication conditions when the sender’s retries are not set to 0. Toggles between Enabled and Disabled. | Enabled |
| Error Detection | Toggles between CRC and BCC. | CRC |
| RTS Off Delay | Specifies the delay time between when the last serial character is sent to the modem and when RTS is deactivated. Gives modem extra time to transmit the last character of a packet. The valid range is 0–255 and can be set in increments of 5 ms. | 0 |
| RTS Send Delay | Specifies the time delay between setting RTS (request to send) until checking for the CTS (clear to send) response. For use with modems that are not ready to respond with CTS immediately upon receipt of RTS. The valid range is 0–255 and can be set in increments of 5 ms. | 0 |
| Poll Timeout | Poll Timeout only applies when a slave device initiates a MSG instruction. It is the amount of time that the slave device waits for a poll from the master device. If the slave device does not receive a poll within the Poll Timeout, a MSG instruction error is generated and the ladder program needs to requeue the MSG instruction. The valid range is 0–65535 and can be set in increments of 20 ms. If you are using a MSG instruction, it is recommended that a Poll Timeout value of zero not be used. Poll Timeout is disabled if set to zero. | 3000 (60s) |
| Pre-send Time Delay | Delay time before transmission. Required for 1761-NET-AIC physical half-duplex networks. The 1761-NET-AIC needs delay time to change from transmit to receive mode. The valid range is 0–255 and can be set in increments of 5 ms. | 0 |
| Message Retries | Specifies the number of times a slave device attempts to resend a message packet when it does not receive an ACK from the master device. For use in noisy environments where message packets may become corrupted in transmission. The valid range is 0–255. | 3 |
| EOT Suppression | Slave does not respond when polled if no message is queued. Saves modem transmission power when there is no message to transmit. Toggles between Yes and No. | No |



Considerations When Communicating as a DF1 Slave on a Multi-drop Link

When communication is between either your programming software and a MicroLogix 1000 Programmable Controller or between two MicroLogix Programmable Controllers via a slave-to-slave connection on a larger multi-drop link, the devices depend on a DF1 Master to give each of them polling permission to transmit in a timely manner. As the number of slaves increases on the link (up to 254), the time between when your programming software or the MicroLogix Controller is polled also increases. This increase in time may become larger if you are using low baud rates.

As these time periods grow, the following values may need to be changed to avoid loss of communication:

- programming software - increase poll timeout and reply timeout values
- MicroLogix Programmable Controller - increase poll timeout

Ownership Timeout

When a program download sequence is started by a software package to download a ladder logic program to a MicroLogix controller, the software takes “file ownership” of the processor. File ownership prevents other devices from reading from or writing to the processor while the download is in process. If the controller were to respond to a device’s read commands during the download, the processor could respond with incorrect information. Similarly, if the controller were to accept information from other devices, the information could be lost because the program download sequence could immediately overwrite the information. Once the download is completed, the programming software returns the file ownership to the controller, so other devices can communicate with it again.

With the addition of DF1 half-duplex slave protocol, the controller clears the file ownership if no supported commands are received from the owner within the timeout period. If the file ownership were not cleared after a download sequence interruption, the processor would not accept commands from any other devices because it would assume another device still had file ownership.

If a download sequence is interrupted due to noise caused by electromagnetic interference, discontinue communications to the controller for the *ownership timeout* period and restart the program download. The *ownership timeout* period is set to 60 seconds as a default for all protocols. However, if you are using DF1 half-duplex, and the *poll timeout* value is set to greater than 60 seconds, the *poll timeout* value is used instead of the *ownership timeout*. After the timeout, you can re-establish communications with the processor and try the program download again. The only other way to clear file ownership is to cycle power on the processor.

Using Modems with MicroLogix 1000 Programmable Controllers

The types of modems that you can use with MicroLogix 1000 controllers include dial-up phone modems, leased-line modems, radio modems, and line drivers. For point-to-point full-duplex modem connections that do not require any modem handshaking signals to operate, use DF1 full-duplex protocol. For point-to-multipoint modem connections or for point-to-point modem connections that require Request-to-Send/Clear-To-Send (RTS/CTS) handshaking, use DF1 half-duplex slave protocol. In this case, one (and only one) of the other devices must be configured for DF1 half-duplex master protocol. Do not attempt to use DH-485 protocol through modems under any circumstance.

Important: Only Series D or later MicroLogix 1000 discrete controllers and all MicroLogix 1000 analog controllers support RTS/CTS modem handshaking and only when configured for DF1 half-duplex slave protocol with the control line parameter set to “Half-Duplex Modem”. No other modem handshaking lines (i.e. Data Set Ready, Carrier Detect and Data Terminal Ready) are supported by any MicroLogix 1000 controllers.

Dial-Up Phone Modems

Dial-up phone line modems support point-to-point full-duplex communications. Normally a MicroLogix 1000 controller, on the receiving end of the dial-up connection, is configured for DF1 full-duplex protocol. The modem connected to the MicroLogix 1000 controller must support auto-answer and must not require any modem handshaking signals from the MicroLogix 1000 (i.e., DTR or RTS) in order to operate. The MicroLogix 1000 has no means to cause its modem to initiate or disconnect a phone call, so this must be done from the site of the remote modem.

Leased-Line Modems

Leased-line modems are used with dedicated phone lines that are typically leased from the local phone company. The dedicated lines may be in a point-to-point topology supporting full-duplex communications between two modems or in a point-to-multipoint topology supporting half-duplex communications between three or more modems. In the point-to-point topology, configure the MicroLogix 1000 controllers for DF1 full-duplex protocol (as long as the modems used do not require DTR or RTS to be high in order to operate). In the point-to-multipoint topology, configure the MicroLogix 1000 controllers for DF1 half-duplex slave protocol with the control line parameter set to “Half-Duplex Modem”.

Radio Modems

Radio modems may be implemented in a point-to-point topology supporting either half-duplex or full-duplex communications, or in a point-to-multipoint topology supporting half-duplex communications between three or more modems. In the point-to-point topology using full-duplex radio modems, configure the MicroLogix 1000 controllers for DF1 full-duplex protocol (as long as the modems used do not require DTR or RTS to be high in order to operate). In the point-to-point topology using half-duplex radio modems, or point-to-multipoint topology using half-duplex radio modems, configure the MicroLogix 1000 controllers for DF1 half-duplex slave protocol. If these radio modems require RTS/CTS handshaking, configure the control line parameter to “Half-Duplex Modem”.

Line Drivers

Line drivers, also called short-haul “modems”, do not actually modulate the serial data, but rather condition the electrical signals to operate reliably over long transmission distances (up to several miles). Allen-Bradley’s AIC+ Advanced Interface Converter is a line driver that converts an RS-232 electrical signal into an RS-485 electrical signal, increasing the signal transmission distance from 50 to 4000 feet. In a point-to-point line driver topology, configure the MicroLogix 1000 controller for DF1 full-duplex protocol (as long as the line drivers do not require DTR or RTS to be high in order to operate). In a point-to-multipoint line driver topology, configure the MicroLogix 1000 controllers for DF1 half-duplex slave protocol. If these line drivers require RTS/CTS handshaking, configure the control line parameter to “Half-Duplex Modem”.

DH-485 Communication Protocol

The information in this section describes the DH-485 network functions, network architecture, and performance characteristics. It also helps you plan and operate the MicroLogix 1000 on a DH-485 network.

Important: Only Series C or later MicroLogix 1000 discrete controllers and all MicroLogix 1000 analog controllers support the DH-485 network.

DH-485 Network Description

The DH-485 protocol defines the communication between multiple devices that co-exist on a single pair of wires. This protocol uses RS-485 half-duplex as its physical interface. (RS-485 is a definition of electrical characteristics; it is *not* a protocol.) RS-485 uses devices that are capable of co-existing on a common data circuit, thus allowing data to be easily shared between devices.

The DH-485 network offers:

- interconnection of 32 devices
- multi-master capability
- token passing access control
- the ability to add or remove nodes without disrupting the network
- maximum network length of 1219 m (4000 ft)

The DH-485 protocol supports two classes of devices: initiators and responders. All initiators on the network get a chance to initiate message transfers. To determine which initiator has the right to transmit, a token passing algorithm is used.

The following section describes the protocol used to control message transfers on the DH-485 network.

DH-485 Token Rotation

A node holding the token can send any valid packet onto the network. Each node is allowed only one transmission (plus two retries) each time it receives the token. After a node sends one message packet, it attempts to give the token to its successor by sending a “token pass” packet to its successor.

If no network activity occurs, the initiator sends the token pass packet again. After two retries (a total of three tries), the initiator attempts to find a new successor.

The allowable range of the node address of an initiator is 0 to 31. The allowable address range for all responders is 1 to 31. There must be at least one initiator on the network.

DH-485 Configuration Parameters

When the system mode driver is DH-485 Master, the following parameters can be changed:

| Parameter | Description | Default |
|-------------------|---|---------|
| Baud Rate | Toggles between the communication rate of 9600 and 19200. | 19200 |
| Node Address | This is the node address of the processor on the DH-485 network. The valid range is 1–31. | 1 |
| Max Node Address | This is the maximum node address of an active processor. (fixed at 31) Set the node addresses of the devices on the network to low, sequential numbers for best network performance. Then, set the maximum node address to the value of the last node. | 31 |
| Token Hold Factor | Determines the number of transactions allowed to make each DH-485 token rotation. (fixed at 1) | 1 |

DH-485 Network Initialization

Network initialization begins when a period of inactivity exceeding the time of a link dead timeout is detected by an initiator on the network. When the time for a link dead timeout is exceeded, usually the initiator with the lowest address claims the token. When an initiator has the token, it begins to build the network. The network requires at least one initiator to initialize it.

Building a network begins when the initiator that claimed the token tries to pass the token to the successor node. If the attempt to pass the token fails, or if the initiator has no established successor (for example, when it powers up), it begins a linear search for a successor starting with the node above it in the addressing.

When the initiator finds another active initiator, it passes the token to that node, which repeats the process until the token is passed all the way around the network to the first node. At this point, the network is in a state of normal operation.

Devices that use the DH-485 Network

In addition to the Series C or later MicroLogix 1000 discrete controllers and all MicroLogix 1000 analog controllers, the devices shown in the following table also support the DH-485 network.

Important: You cannot connect the Hand-Held Programmer, 1761-HHP-B30, to the AIC+.

| Catalog Number | Description | Installation Requirement | Function | Publication |
|--|---|---|--|-------------------------------------|
| 1747-L511, -L514, -L524, -L531, -L532 -L541, -L542, -L543 -L551, -L552 -L553 | SLC 500 Processors | SLC Chassis | These processors support a variety of I/O requirements and functionality. | 1747-6.2 |
| 1746-BAS | BASIC Module | SLC Chassis | Provides an interface for SLC 500 devices to foreign devices. Program in BASIC to interface the 3 channels (2 RS232 and 1 DH-485) to printers, modems, or the DH-485 network for data collection. | 1746-6.1 1746-6.2 1746-6.3 |
| 1785-KA5 | DH+™/DH-485 Gateway | (1771) PLC Chassis | Provides communication between stations on the PLC-5® (DH+) and SLC 500 (DH-485) networks. Enables communication and data transfer from PLC® to SLC 500 on DH-485 network. Also enables programming software programming or data acquisition across DH+ to DH-485. | 1785-6.5.5 1785-1.21 |
| 2760-RB | Flexible Interface Module | (1771) PLC Chassis | Provides an interface for SLC 500 (using protocol cartridge 2760-SFC3) to other A-B PLCs and devices. Three configurable channels are available to interface with Bar Code, Vision, RF, Dataliner™, and PLC systems. | 2760-ND001 |
| 1784-KTX, -KTXD | PC DH-485 IM | IBM XT/AT Computer Bus | Provides DH-485 using RSLinx | 1784-6.5.22 |
| 1784-PCMK | PCMCIA IM | PCMCIA slot in computer and Interchange | Provides DH-485 using RSLinx | 1784-6.5.19 |
| 1747-PT1 | Hand-Held Terminal | NA | Provides hand-held programming, monitoring, configuring, and troubleshooting capabilities for SLC 500 processors. | 1747-NP002 |
| 1747-DTAM, 2707-L8P1, -L8P2, -L40P1, -L40P2, -V40P1, -V40P2, -V40P2N, -M232P3, and -M485P3 | DTAM, DTAM Plus, and DTAM Micro Operator Interfaces | Panel Mount | Provides electronic operator interface for SLC 500 processors. | 1747-ND013 2707-800, 2707-803 |
| 2711-K5A2, -B5A2, -K5A5, -B5A5, -K5A1, -B5A1, -K9A2, -T9A2, -K9A5, -T9A5, -K9A1, and -T9A1 | PanelView 550 and PanelView 900 Operator Terminals | Panel Mount | Provides electronic operator interface for SLC 500 processors. | 2711-802, 2711-816 |

NA = Not Applicable

Important DH-485 Network Planning Considerations

Carefully plan your network configuration before installing any hardware. Listed below are some of the factors that can affect system performance:

- amount of electrical noise, temperature, and humidity in the network environment
- number of devices on the network
- connection and grounding quality in installation
- amount of communication traffic on the network
- type of process being controlled
- network configuration

The major hardware and software issues you need to resolve before installing a network are discussed in the following sections.

Hardware Considerations

You need to decide the length of the communication cable, where you route it, and how to protect it from the environment where it will be installed.

When the communication cable is installed, you need to know how many devices are to be connected during installation and how many devices will be added in the future. The following sections help you understand and plan the network.

Number of Devices and Length of Communication Cable

You must install an AIC+ Advanced Interface Converter, catalog number 1761-NET-AIC, for each node on the network. If you plan to add nodes later, provide additional advanced interface converters during the initial installation to avoid recabling after the network is in operation.

The maximum length of the communication cable is 1219 m (4000 ft). This is the total cable distance from the first node to the last node on the network.

Planning Cable Routes

Follow these guidelines to help protect the communication cable from electrical interference:

- Keep the communication cable at least 1.52 m (5 ft) from any electric motors, transformers, rectifiers, generators, arc welders, induction furnaces, or sources of microwave radiation.
- If you must run the cable across power feed lines, run the cable at right angles to the lines.
- If you do not run the cable through a contiguous metallic wireway or conduit, keep the communication cable at least 0.15 m (6 in.) from ac power lines of less than 20A, 0.30 m (1 ft) from lines greater than 20A, but only up to 100k VA, and 0.60 m (2 ft) from lines of 100k VA or more.
- If you run the cable through a contiguous metallic wireway or conduit, keep the communication cable at least 0.08 m (3 in.) from ac power lines of less than 20A, 0.15 m (6 in.) from lines greater than 20A, but only up to 100k VA, and 0.30 m (1 ft) from lines of 100k VA or more.

Running the communication cable through conduit provides extra protection from physical damage and electrical interference. If you route the cable through conduit, follow these additional recommendations:

- Use ferromagnetic conduit near critical sources of electrical interference. You can use aluminum conduit in non-critical areas.
- Use plastic connectors to couple between aluminum and ferromagnetic conduit. Make an electrical connection around the plastic connector (use pipe clamps and the heavy gauge wire or wire braid) to hold both sections at the same potential.
- Ground the entire length of conduit by attaching it to the building earth ground.
- Do not let the conduit touch the plug on the cable.
- Arrange the cables loosely within the conduit. The conduit should contain only serial communication cables.
- Install the conduit so that it meets all applicable codes and environmental specifications.

For more information on planning cable routes, see *Industrial Automation Wiring and Grounding Guidelines*, Publication Number 1770-4.1.

Software Considerations

Software considerations include the configuration of the network and the parameters that can be set to the specific requirements of the network. The following are major configuration factors that have a significant effect on network performance:

- number of nodes on the network
- addresses of those nodes
- baud rate

The following sections explain network considerations and describe ways to select parameters for optimum network performance (speed). See your programming software's user manual for more information.

Number of Nodes

The number of nodes on the network directly affects the data transfer time between nodes. Unnecessary nodes (such as a second programming terminal that is not being used) slow the data transfer rate. The maximum number of nodes on the network is 32.

Setting Node Addresses

The best network performance occurs when node addresses are assigned in sequential order. Initiators, such as personal computers, should be assigned the lowest numbered addresses to minimize the time required to initialize the network. The valid range for the MicroLogix 1000 controllers is 1–31 (controllers cannot be node 0). The default setting is 1. The node address is stored in the controller status file (S:16L).

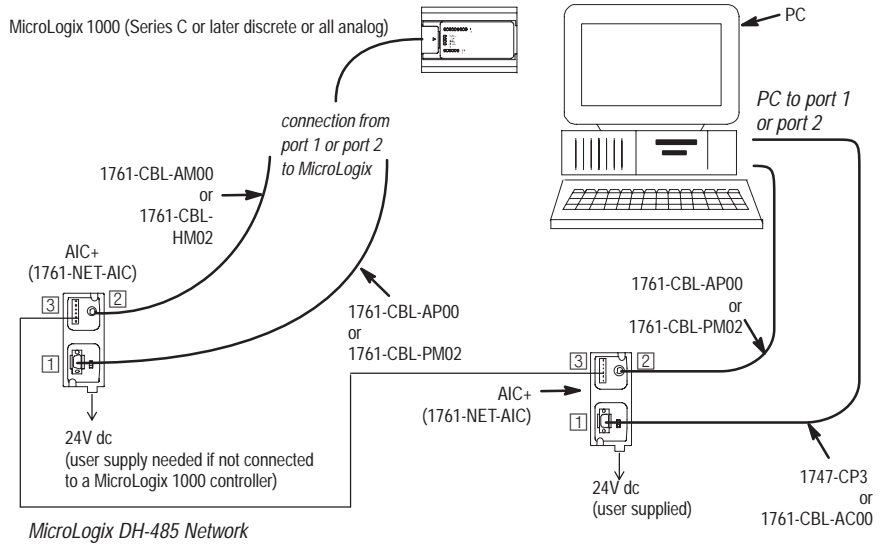
Setting Controller Baud Rate

The best network performance occurs at the highest baud rate, which is 19200. This is the default baud rate for a MicroLogix 1000 device on the DH-485 network. All devices must be at the same baud rate. This rate is stored in the controller status file (S:16H).

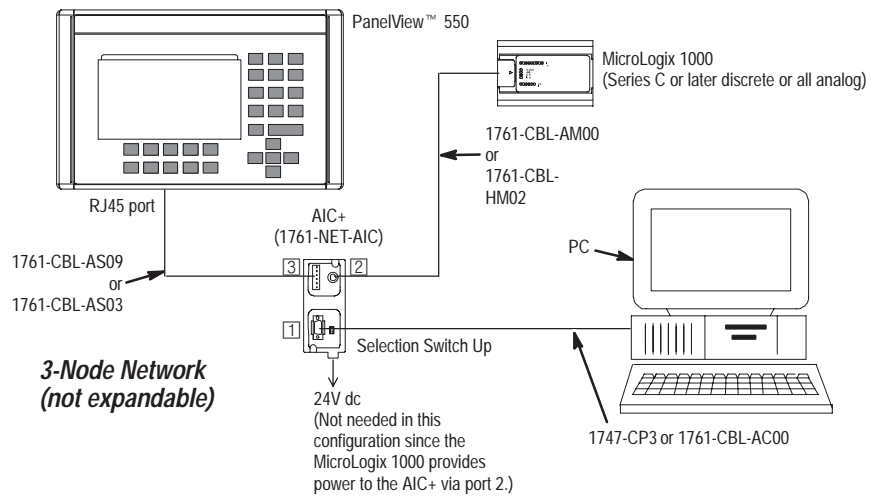
Example DH-485 Connections

The following network diagrams provide examples of how to connect Series C or later MicroLogix 1000 discrete and all MicroLogix 1000 analog controllers to the DH-485 network using the AIC+. For more information on the AIC+, see the *Advanced Interface Converter and DeviceNet Interface Installation Instructions*, Publication 1761-5.11.

DH-485 Network with a MicroLogix 1000 Controller

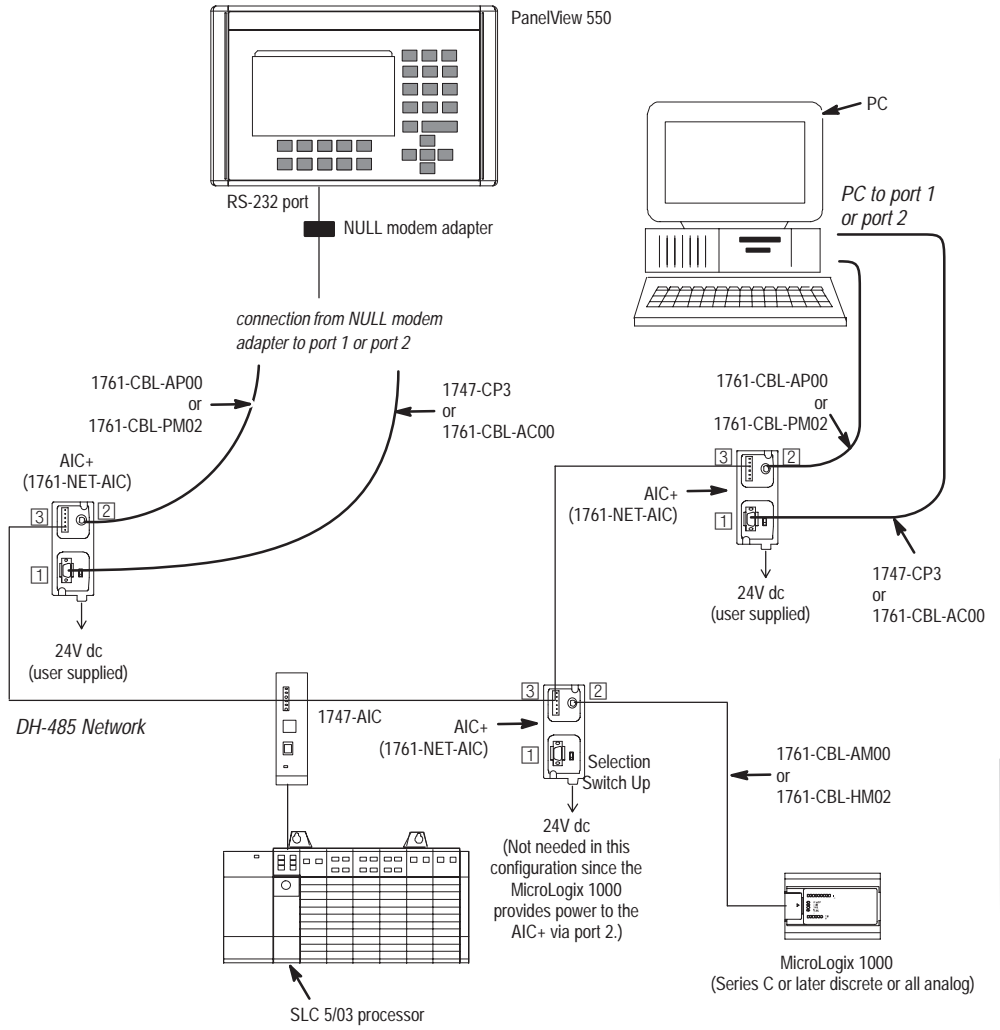


Typical 3-Node Network



- 1 DB-9 RS-232 port
- 2 mini-DIN 8 RS-232 port
- 3 DH-485/DF1 port

Networked Operator Interface Device and MicroLogix Controller



- 1 DB-9 RS-232 port
- 2 mini-DIN 8 RS-232 port
- 3 DH-485/DF1 port

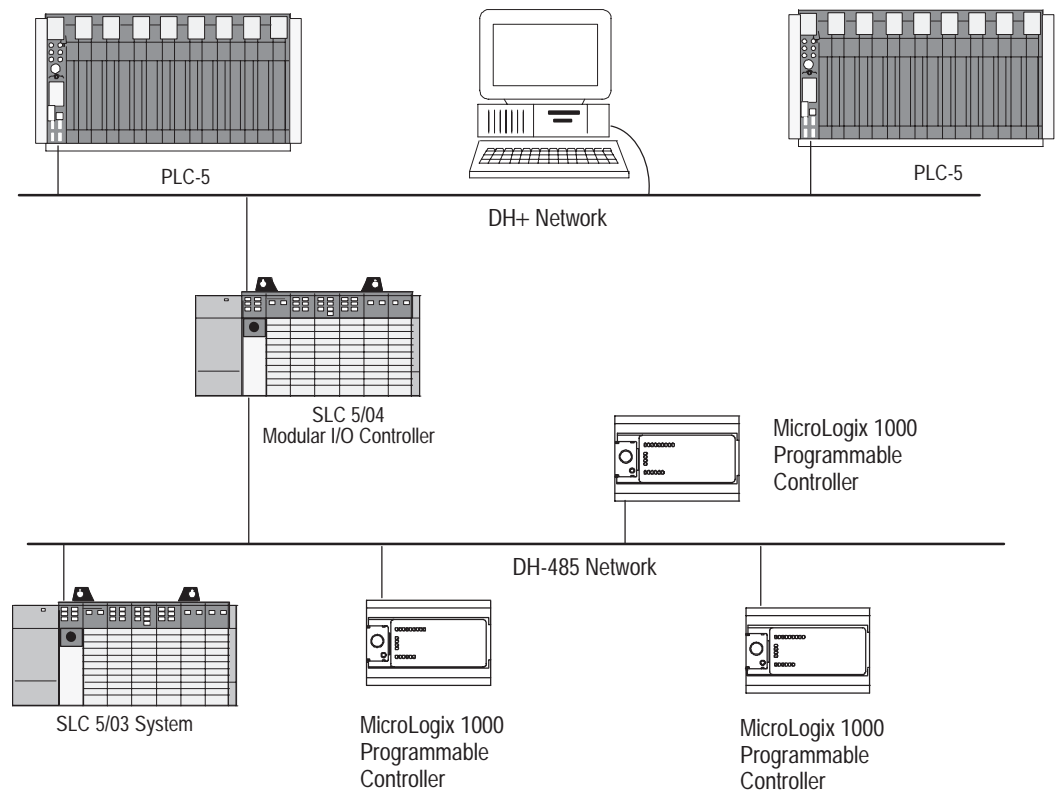
MicroLogix Remote Packet Support

Series D MicroLogix discrete controllers and all MicroLogix analog controllers can respond to communication packets (or commands) that do not originate on the local DH-485 network. This is useful in installations where communication is needed between the DH-485 and DH+ networks.

The example below shows how to send messages from a PLC device or a PC on the DH+ network to a MicroLogix 1000 controller on the DH-485 network. This method uses an SLC 5/04 processor bridge connection.

When using this method:

- PLC-5 devices can send read and write commands to MicroLogix controllers.
- MicroLogix 1000 controllers can respond to MSG instructions received. The MicroLogix controllers cannot initiate MSG instructions to devices on the DH+ network.
- PC can send read and write commands to MicroLogix controllers.
- PC can do remote programming of MicroLogix controllers.



Application Example Programs

This appendix is designed to illustrate various instructions described previously in this manual. Application example programs include:

- paper drilling machine using most of the software instructions
- time-driven sequencer using TON and SQO instructions
- event-driven sequencer using SQC and SQO instructions
- bottle line example using the HSC instruction (up/down counter)
- pick and place machine example using the HSC instruction (Quadrature encoder with reset and hold)
- RPM calculation using HSC, RTO, timer, and math instructions
- on/off circuit using basic, program flow, and application specific instructions
- spray booth using bit shift and FIFO instructions
- adjustable time delay example using timer instructions

Because of the variety of uses for this information, the user of and those responsible for applying this information must satisfy themselves as to the acceptability of each application and use of the program. In no event will Allen-Bradley Company be responsible or liable for indirect or consequential damages resulting from the use of application of this information.

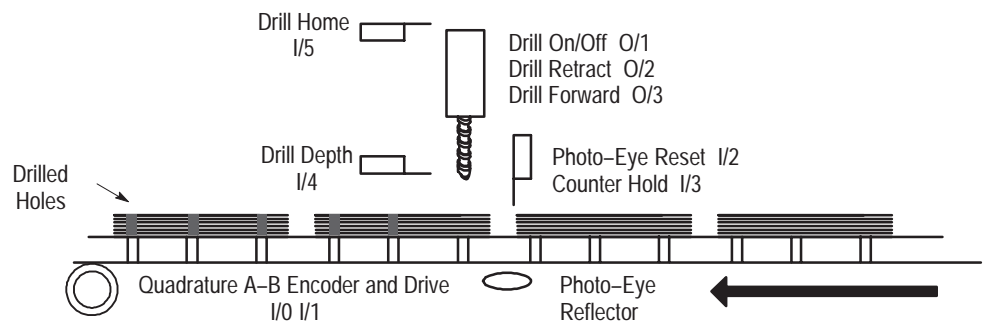
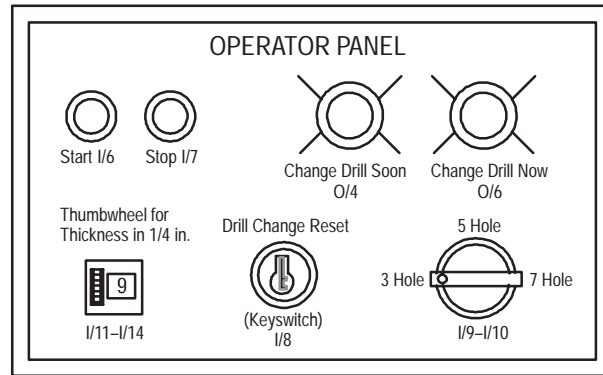
The illustrations, charts, and examples shown in this appendix are intended solely to illustrate the principles of the controller and some of the methods used to apply them. Particularly because of the many requirements associated with any particular installation, Allen-Bradley Company cannot assume responsibility or liability for actual use based upon the illustrative uses and applications.

Paper Drilling Machine Application Example

For a detailed explanation of:

- LD, LDI, OUT, RES, SET, RST, and OSR instructions, see chapter 8.
- EQU and GEQ instructions, see chapter 9.
- CLR, ADD, and SUB instructions, see chapter 10.
- MOV and FRD instructions, see chapter 11.
- JSR and RET instructions, see chapter 12.
- INT and SQO instructions, see chapter 13.
- HSC, HSL, and RAC instructions, see chapter 14.

This machine can drill three different hole patterns into bound manuals. The program tracks drill wear and signals the operator that the bit needs replacement. The machine shuts down if the signal is ignored by the operator.



Conveyor Enable wired in series to the Drive O/5
Conveyor Drive Start/Stop wired in series to the Drive O/0

20226

Paper Drilling Machine Operation Overview

Undrilled books are placed onto a conveyor taking them to a single spindle drill. Each book moves down the conveyor until it reaches the first drilling position. The conveyor stops moving and the drill lowers and drills the first hole. The drill then retracts and the conveyor moves the same book to the second drilling position. The drilling process is repeated until there are the desired holes per book.

Drill Mechanism Operation

When the operator presses the start button, the drill motor turns on. After the book is in the first drilling position, the conveyor subroutine sets a drill sequence start bit and the drill moves toward the book. When the drill has drilled through the book, the drill body hits a limit switch and causes the drill to retract up out of the book. When the drill body is fully retracted, the drill body hits another limit switch indicating that it is in the home position. Hitting the second limit switch unlatches the drill sequence start bit and causes the conveyor to move the book to the next drilling position.

Conveyor Operation

When the start button is pressed, the conveyor moves the books forward. As the first book moves close to the drill, the book trips a photo-eye sensor. This tells the machine where the leading edge of the book is. Based on the position of the selector switch, the conveyor moves the book until it reaches the first drilling position. The drill sequence start bit is set and the first hole is drilled. The drill sequence start bit is now unlatched and the conveyor moves the same book to the second drilling position. The drilling process is repeated until there are the desired holes per book. The machine then looks for another book to break the photo-eye beam and the process is repeated. The operator can change the number of drilled holes by changing the selector switch.

Drill Calculation and Warning

The program tracks the number of holes drilled and the number of inches of material that have been drilled through using a thumbwheel. The thumbwheel is set to the thickness of the book per 1/4 inch. (If the book is 1 1/2 inches thick, the operator would set the thumbwheel to 6.) When 25,000 inches have been drilled, the Change Drill Soon pilot light turns on. When 25,500 inches have been drilled, the Change Drill Soon pilot light flashes. When 26,000 inches have been drilled, the Change Drill Now pilot light turns on and the machine turns off. The operator changes drill bits and then resets the internal drill wear counter by turning the Drill Change Reset keyswitch.

Paper Drilling Machine Ladder Program

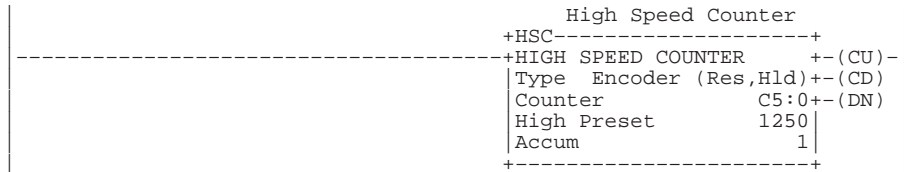
Rung 2:0

Initializes the high-speed counter each time the RRUN mode is entered. The high-speed counter data area (N7:5 - N7:9) corresponds with the starting address (source address) of the HSL instruction. The HSC instruction is disabled each entry into the RRUN mode until the first time that it is executed as true. (The high preset was "pegged" on initialization to prevent a high preset interrupt from occurring during the initialization process.)

| | |
|---------------|--|
| 1'st Pass | Output Mask (only use bit 0 ie. 0:0/0) |
| S:1 | +MOV-----+ |
| -----] [----- | +MOVE-----+ |
| 15 | Source 1 |
| | Dest N7:5 0 |
| | +-----+ |
| | High Output Pattern (turn off 0:0/0) |
| | +MOV-----+ |
| | +MOVE-----+ |
| | Source 0 |
| | Dest N7:6 0 |
| | +-----+ |
| | High Preset Value (counts to next hole) |
| | +MOV-----+ |
| | +MOVE-----+ |
| | Source 32767 |
| | Dest N7:7 0 |
| | +-----+ |
| | Low output pattern (turn on 0:0/0 each reset) |
| | +MOV-----+ |
| | +MOVE-----+ |
| | Source 1 |
| | Dest N7:8 0 |
| | +-----+ |
| | Low preset value (cause low preset int at reset) |
| | +MOV-----+ |
| | +MOVE-----+ |
| | Source 0 |
| | Dest N7:9 0 |
| | +-----+ |
| | High Speed Counter |
| | +HSL-----+ |
| | +HSC LOAD-----+ |
| | Counter C5:0 |
| | Source N7:5 |
| | Length 5 |
| | +-----+ |

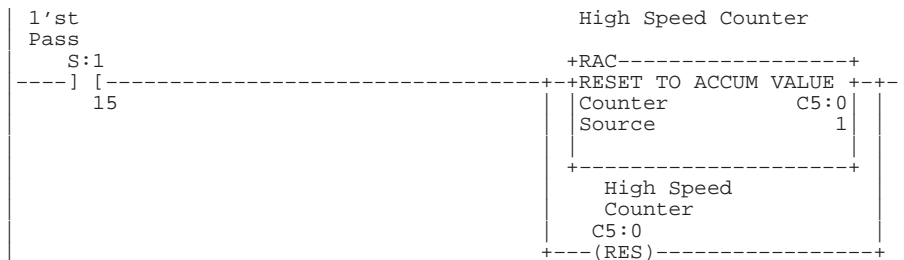
Rung 2:1

This HSC instruction is not placed in the high-speed counter interrupt subroutine. If this instruction were placed in the interrupt subroutine, the high-speed counter could never be started or initialized (because an interrupt must first occur in order to scan the high-speed counter interrupt subroutine).



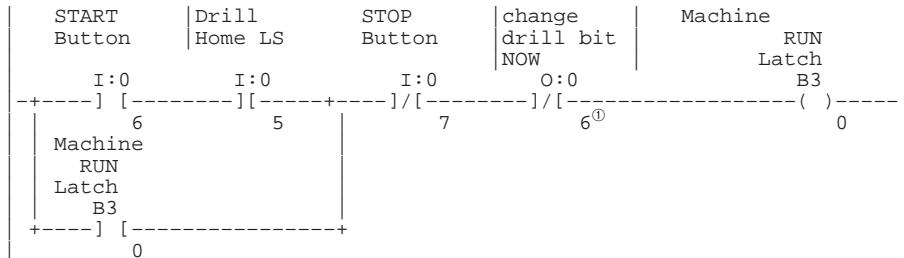
Rung 2:2

Forces a high-speed counter low preset interrupt to occur each RRUN mode entry. An interrupt can only occur on the transition of the high-speed counter accum to a preset value (accum reset to 1, then 0). This is done to allow the high-speed counter interrupt subroutine sequencers to initialize. The order of high-speed counter initialization is: (1)load high-speed counter parameters (2)execute HSL instruction (3)execute true HSC instruction (4)(optional) force high-speed counter interrupt to occur.



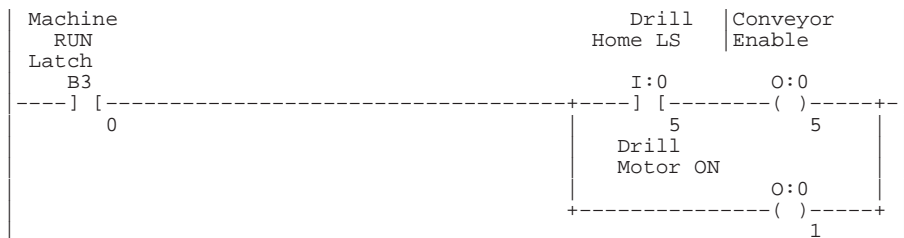
Rung 2:3

Starts the conveyor in motion when the start button is pressed. However, another condition must also be met before we start the conveyor: the drill bit must be in its fully retracted position (home). This rung also stops the conveyor when the stop button is pressed.



Rung 2:4

Applies the above start logic to the conveyor and drill motor.



① This instruction accesses I/O only available with 32 I/O controllers. Therefore, do not include this instruction if you are using a 16 I/O controller.

Rung 2:5
Calls the drill sequence subroutine. This subroutine manages the operation of a drilling sequence and restarts the conveyor upon completion of the drilling sequence.

```

+JSR-----+
-----+JUMP TO SUBROUTINE+
|SBR file number 6|
+-----+

```

Rung 2:6
Calls the subroutine that tracks the amount of wear on the current drill bit.

```

+JSR-----+
-----+JUMP TO SUBROUTINE+
|SBR file number 7|
+-----+

```

Rung 2:7

```

-----+END+-----

```

Rung 4:0
Resets the hole count sequencers each time that the low preset is reached. The low preset has been set to zero to cause an interrupt to occur each time that a reset occurs. The low preset is reached anytime that a reset C5:0 or hardware reset occurs. This ensures that the first preset value is loaded into the high-speed counter at each entry into the RRUN mode and each time that the external reset signal is activated.

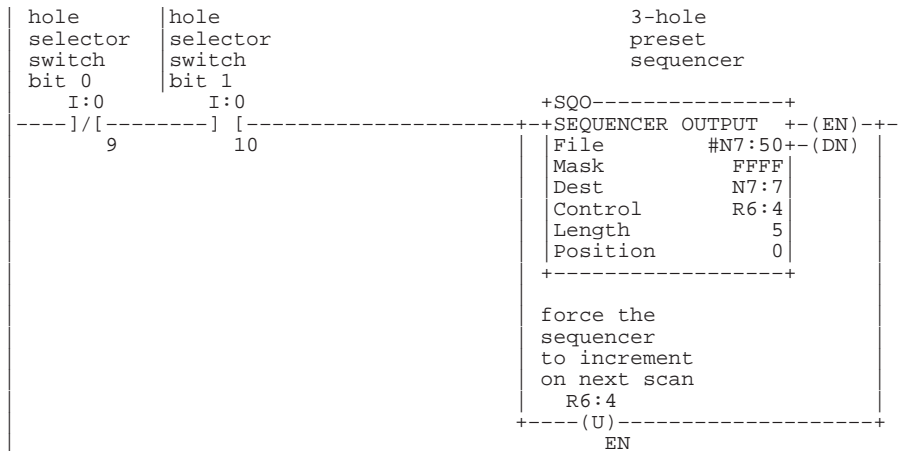
```

                                     interrupt
                                     occurred
                                     due to
                                     low preset
                                     reached
+INT-----+ C5:0                                     R6:4
+INTERRUPT SUBROUTINE +----] [-----] +----(RES)-----+
+-----+ IL
                                     5-hole
                                     preset
                                     sequencer
                                     R6:5
                                     +----(RES)-----+
                                     7-hole
                                     preset
                                     sequencer
                                     R6:6
                                     +----(RES)-----+

```

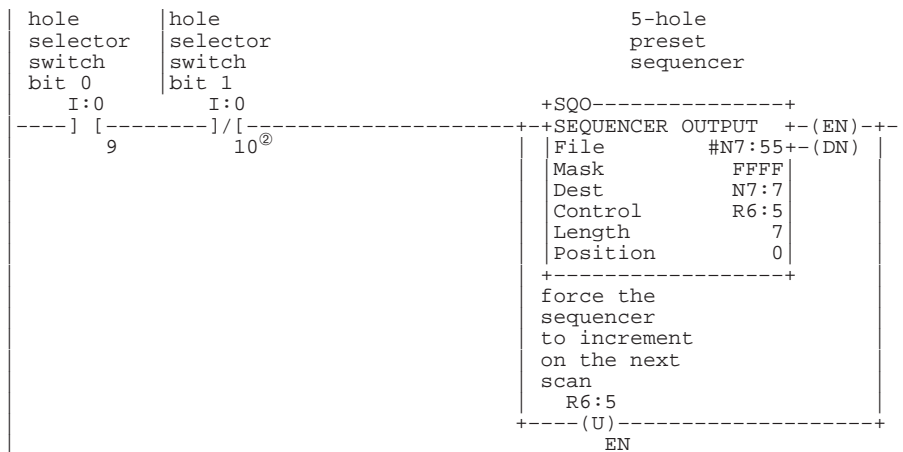
Rung 4:1^①

Keeps track of the hole number that is being drilled and loads the correct high-speed counter preset based on the hole count. This rung is only active when the "hole selector switch" is in the "3-hole" position. The sequencer uses step 0 as a null step upon reset. It uses the last step as a "go forever" in anticipation of the "end-of-manual" hard-wired external reset.



Rung 4:2

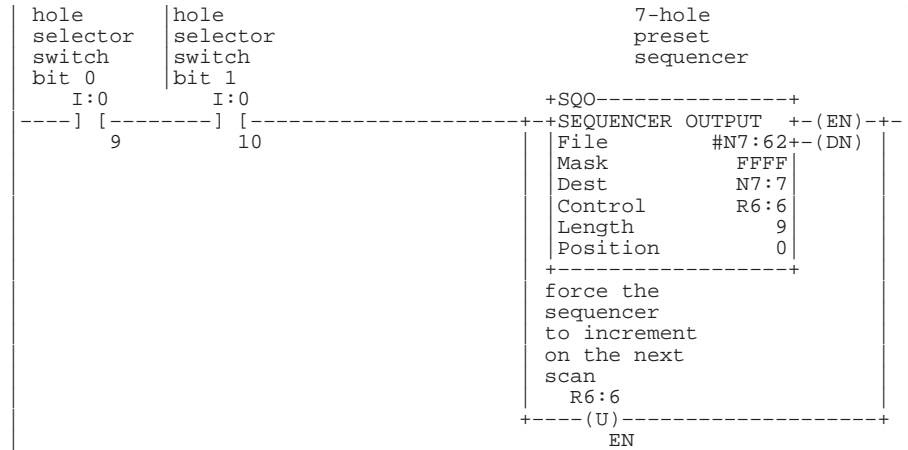
Is identical to the previous rung except that it is only active when the "hole selector switch" is in the "5-hole" position.



- ① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.
- ② This instruction accesses I/O only available with 32 I/O controllers. Therefore, do not include this instruction if you are using a 16 I/O controller.

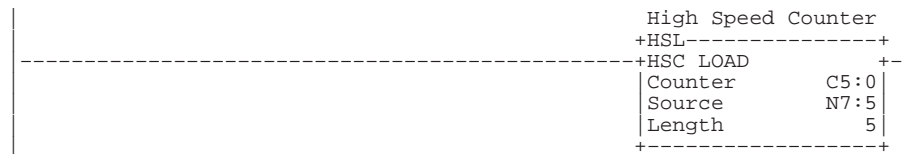
Rung 4:3^①

Is identical to the two previous rungs except that it is only active when the "hole selector switch" is in the "7-hole" position.



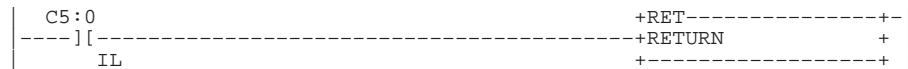
Rung 4:4

Ensures that the high-speed counter preset value (N7:7) is immediately applied to the HSC instruction.



Rung 4:5

Interrupt occurred due to low preset reached.

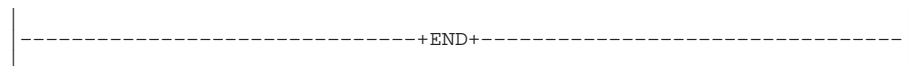


Rung 4:6

Signals the main program (file 2) to initiate a drilling sequence. The high-speed counter has already stopped the conveyor at the correct position using its high preset output pattern data (clear O:0/0). This occurred within microseconds of the high preset being reached (just prior to entering this high-speed counter interrupt subroutine). The drill sequence subroutine resets the drill sequence start bit and sets the conveyor drive bit (O:0/0) upon completion of the drilling sequence.



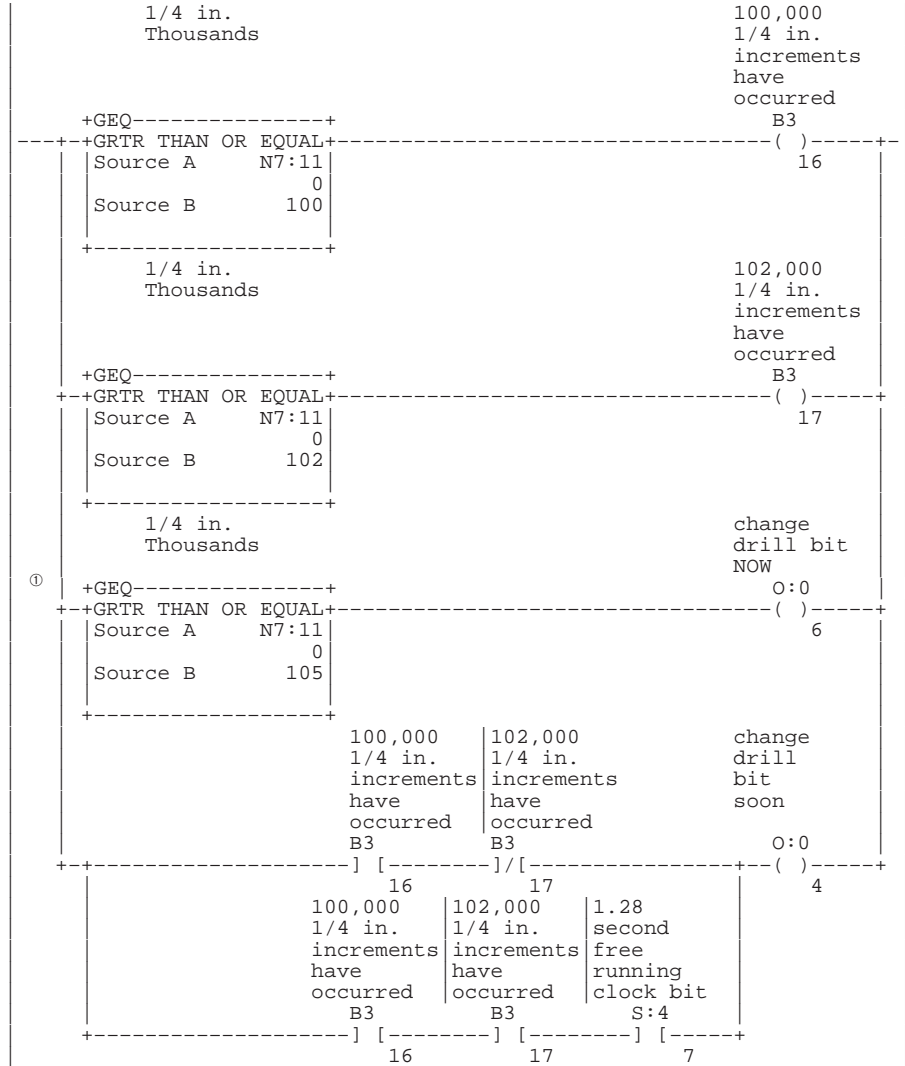
Rung 4:7



① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.

Rung 7:0

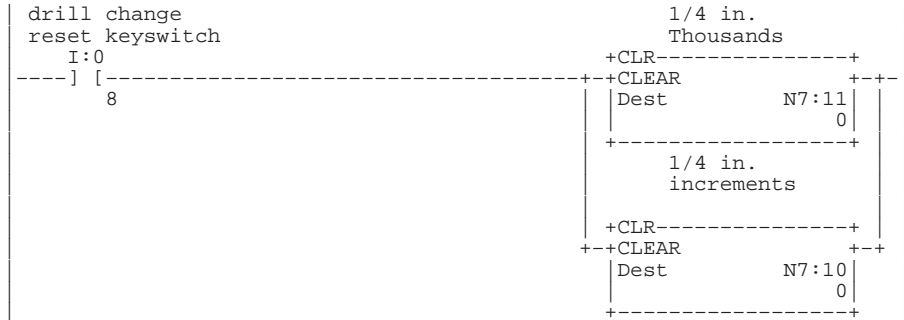
Examines the number of 1/4 in. thousands that have accumulated over the life of the current drill bit. If the bit has drilled between 100,000-101,999 1/4 in. increments of paper, the "change drill" light illuminates steadily. When the value is between 102,000-103,999, the "change drill" light flashes at a 1.28 second rate. When the value reaches 105,000, the "change drill" light flashes and the "change drill now" light illuminates.



① This branch accesses I/O only available with 32 I/O controllers. Therefore, do not include this branch if you are using a 16 I/O controller.

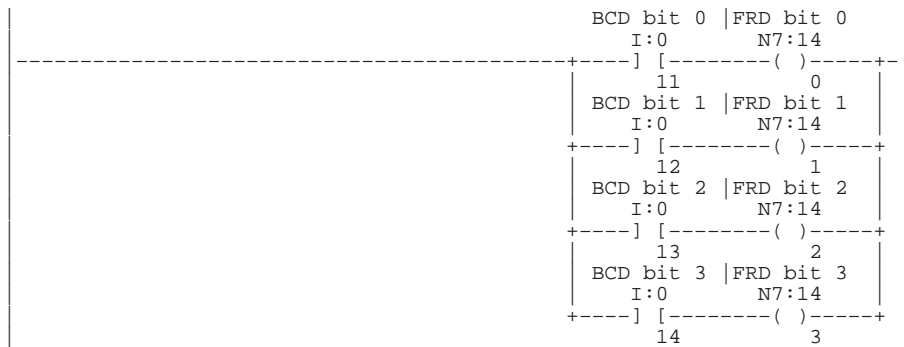
Rung 7:1

Resets the number of 1/4 in. increments and the 1/4 in. thousands when the "drill change reset" keyswitch is energized. This should occur following each drill bit change.



Rung 7:2^①

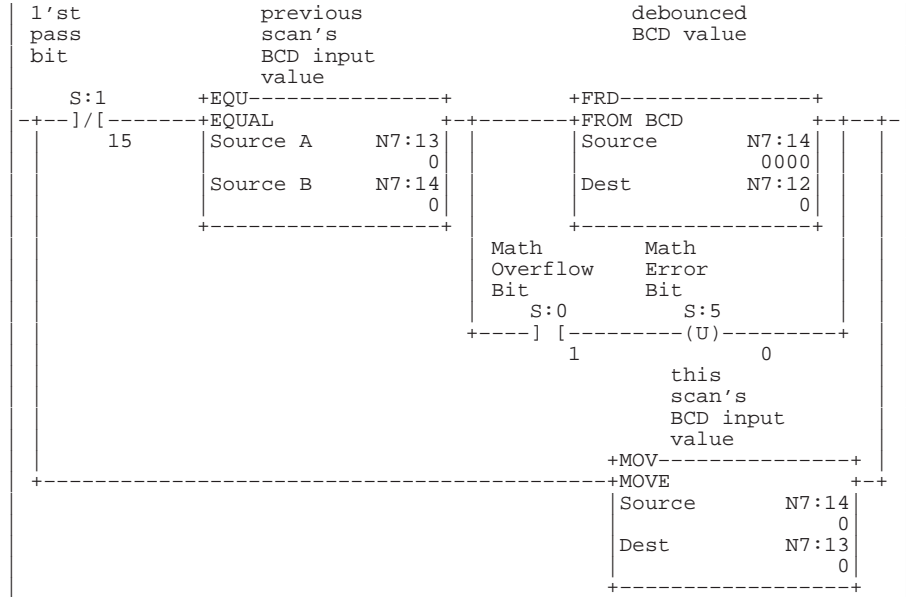
Moves the single digit BCD thumbwheel value into an internal integer register. This is done to properly align the four BCD input signals prior to executing the BCD to Integer instruction (FRD). The thumbwheel is used to allow the operator to enter the thickness of the paper that is to be drilled. The thickness is entered in 1/4 in. increments. This provides a range of 1/4 in. to 2.25 in.



① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.

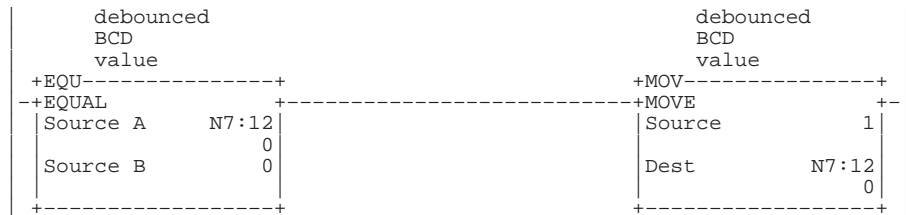
Rung 7:3

Converts the BCD thumbwheel value from BCD to integer. This is done because the controller operates upon integer values. This rung also "debounces" the thumbwheel to ensure that the conversion only occurs on valid BCD values. Note that invalid BCD values can occur while the operator is changing the BCD thumbwheel. This is due to input filter propagation delay differences between the 4 input circuits that provide the BCD input value.



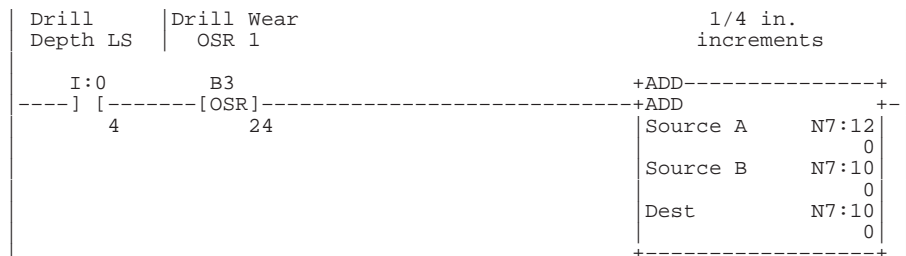
Rung 7:4

Ensures that the operator cannot select a paper thickness of 0. If this were allowed, the drill bit life calculation could be defeated resulting in poor quality holes due to a dull drill bit. Therefore the minimum paper thickness used to calculate drill bit wear is 1/4 in.



Rung 7:5

Keeps a running total of how many inches of paper have been drilled with the current drill bit. Every time a hole is drilled, adds the thickness (in 1/4 ins) to the running total (kept in 1/4 ins). The OSR is necessary because the ADD executes every time the rung is true, and the drill body would actuate the DRILL DEPTH limit switch for more than 1 program scan. Integer N7:12 is the integer-converted value of the BCD thumbwheel on inputs I:0/11 - I:0/14.



Rung 7:6

When the number of 1/4 in. increments surpasses 1000, finds out how many increments are past 1000 and stores in N7:20. Adds 1 to the total of '1000 1/4 in.' increments and re-initializes the 1/4 in. increments accumulator to how many increments were beyond 1000.

```

      1/4 in.
      increments
+GEQ-----+
+GRTR THAN OR EQUAL+-----+
Source A      N7:10
              0
Source B      1000
+-----+
+SUB-----+
+SUBTRACT-----+
Source A      N7:10
              0
Source B      1000
Dest          N7:20
              0
+-----+
      1/4 in.
      Thousands
+ADD-----+
+ADD-----+
Source A      1
Source B      N7:11
              0
Dest          N7:11
              0
+-----+
      1/4 in.
      increments
+MOV-----+
+MOVE-----+
Source        N7:20
              0
Dest          N7:10
              0
+-----+
+-----+-----+-----+
+END+-----+

```

Rung 7:7

Paper Drilling Machine Instruction List Program

File 2, Rung 0

Initializes the high-speed counter each time the RRUN mode is entered. The high-speed counter data area (N7:5 - N7:9) corresponds with the starting address (source address) of the HSL instruction. The HSC instruction is disabled each entry into the RRUN mode until the first time that it is executed as true. (The high preset was "pegged" on initialization to prevent a high preset interrupt from occurring during the initialization process.)

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|----------------|--------|
| 20 | -] [- | LD | 1'st Pass | | |
| | | | S1/15 | 0 | |
| 106 | | MOV | SRC Output Mask (only use bit 0 ie. 0:0/0) DEST N5 | 1 0000H | |
| 106 | | MOV | SRC High Output Pattern (turn off 0:0/0) DEST N6 | 0 0000H | |
| 106 | | MOV | SRC High Preset Value (counts to next hole) DEST N7 | 32767 0000H | |
| 106 | | MOV | SRC Low output pattern (turn on 0:0/0 each reset) DEST N8 | 1 0000H | |
| 106 | | MOV | SRC Low preset value (cause low preset intat reset) DEST N9 | 0 0000H | |
| 171 | | HSL | High Speed Counter CNTR C0 Output Mask (only use bit 0 ie. 0:0/0) SRC N5 LEN | 5 | |

File 2, Rung 1

This HSC instruction is not placed in the high-speed counter interrupt subroutine. If this instruction were placed in the interrupt subroutine, the high-speed counter could never be started or initialized (because an interrupt must first occur in order to scan the high-speed counter interrupt subroutine).

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|------------------------------------|--------|
| 170 | | HSC | TYPE High Speed Counter CNTR C0 PRE ACC | Encoder (Res,Hld) 1250 0000H | |

File 2, Rung 2

Forces a high-speed counter low preset interrupt to occur each RRUN mode entry. An interrupt can only occur on the transition of the high-speed counter accum to a preset value (accum reset to 1, then 0). This is done to allow the high-speed counter interrupt subroutine sequencers to initialize. The order of high-speed counter initialization is: (1)load high-speed counter parameters (2)execute HSL instruction (3)execute true HSC instruction (4)(optional) force high-speed counter interrupt to occur.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--------------------------------------|-------|--------|
| 20 | -] [- | LD | 1'st Pass S1/15 | 0 | |
| 132 | | RAC | High Speed Counter CNTR C0 SRC | 1 | |
| 7 | | RES | High Speed Counter C0 | | |

File 2, Rung 3

Starts the conveyor in motion when the start button is pressed. However, another condition must also be met before we start the conveyor: the drill bit must be in its fully retracted position (home). This rung also stops the conveyor when the stop button is pressed.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-----------------|-------------------|----------|-----------------------------|-------|--------|
| 20 | -] [- | LD | START Button I/6 | 0 | |
| 22 | -] [- | AND | Drill Home LS I/5 | 0 | |
| 24 | _] [_ | OR | Machine RUN Latch B/0 | 0 | |
| 23 | -] / [- | ANI | STOP Button I/7 | 0 | |
| 23 ^① | -] / [- | ANI | change drill bit NOW O/6 | 0 | |
| 40 | - () - | OUT | Machine RUN Latch B/0 | 0 | |

① This instruction accesses I/O only available with 32 I/O controllers. Therefore, do not include this instruction if you are using a 16 I/O controller.

File 2, Rung 4

Applies the above start logic to the conveyor and drill motor.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| 20 | -] [- | LD | Machine RUN Latch B/0 | 0 | |
| 10 | | MPS | | | |
| 22 | -] [- | AND | Drill Home LS I/5 | 0 | |
| 40 | - () - | OUT | Conveyor Enable O/5 | 0 | |
| 12 | | MPP | | | |
| 40 | - () - | OUT | Drill Motor ON O/1 | 0 | |

File 2, Rung 5

Calls the drill sequence subroutine. This subroutine manages the operation of a drilling sequence and restarts the conveyor upon completion of the drilling sequence.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| 132 | | JSR | SBR# | 6 | |

File 2, Rung 6

Calls the subroutine that tracks the amount of wear on the current drill bit.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| 132 | | JSR | SBR# | 7 | |

File 4, Rung 0

Resets the hole count sequencers each time that the low preset is reached. The low preset has been set to zero to cause an interrupt to occur each time that a reset occurs. The low preset is reached anytime that a reset C5:0 or hardware reset occurs. This ensures that the first preset value is loaded into the high-speed counter at each entry into the RRUN mode and each time that the external reset signal is activated.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|-------|--------|
| 158 | -INT- | LD-INT | | | |
| 22 | -] [- | AND | interrupt occurred due to low preset reached C0/IL | 0 | |
| 7 | | RES | 3 hole preset sequencer R4 | | |
| 7 | | RES | 5 hole preset sequencer R5 | | |
| 7 | | RES | 7 hole preset sequencer R6 | | |

File 4, Rung 1^①

Keeps track of the hole number that is being drilled and loads the correct high-speed counter preset based on the hole count. This rung is only active when the "hole selector switch" is in the "3-hole" position. The sequencer uses step 0 as a null step upon reset. It uses the last step as a "go forever" in anticipation of the "end-of-manual" hard-wired external reset.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|-------|--------|
| 21 | -]/[- | LDI | hole selector switch bit 0 I/9 | 0 | |
| 22 | -] [- | AND | hole selector switch bit 1 I/10 | 0 | |
| 152 | | SQO | FILE #N50 MASK | FFFFH | |
| | | | High Preset Value (counts to next hole) DEST N7 | | |
| | | | 3 hole preset sequencer CTRL R4 | | |
| | | | LEN | 5 | |
| | | | POS | 0000H | |
| 42 | -(U)- | RST | force the sequencer to increment on next scan R4/EN | 1 | |

① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.

File 4, Rung 2

Is identical to the previous rung except that it is only active when the "hole selector switch" is in the "5-hole" position.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|------------------|--|-------|--------|
| 20 | -] [- | LD | hole selector switch bit 0 I/9 | 0 | |
| 23 | -] / [- | ANI ^① | hole selector switch bit 1 I/10 | 0 | |
| 152 | | SQO | FILE #N55 MASK | FFFFH | |
| | | | High Preset Value (counts to next hole) DEST N7 5-hole preset sequencer CTRL R5 LEN 7 POS 0000H | | |
| 42 | -(U)- | RST | force the sequencer to increment on the next scan R5/EN | 1 | |

File 4, Rung 3^②

Is identical to the two previous rungs except that it is only active when the "hole selector switch" is in the "7-hole" position.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|-------|--------|
| 20 | -] [- | LD | hole selector switch bit 0 I/9 | 0 | |
| 22 | -] [- | AND | hole selector switch bit 1 I/10 | 0 | |
| 152 | | SQO | FILE #N62 MASK | FFFFH | |
| | | | High Preset Value (counts to next hole) DEST N7 7-hole preset sequencer CTRL R6 LEN 9 POS 0000H | | |
| 42 | -(U)- | RST | force the sequencer to increment on the next scan R6/EN | 1 | |

① This instruction accesses I/O only available with 32 I/O controllers. Therefore, do not include this instruction if you are using a 16 I/O controller.

② This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.

File 4, Rung 4

Ensures that the high-speed counter preset value (N7:7) is immediately applied to the HSC instruction.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|-------|--------|
| 171 | | HSL | High Speed Counter CNTR C0 | | |
| | | | Output Mask (only use bit 0 ie. O:0/0) | | |
| | | | SRC N5 | | |
| | | | LEN | 5 | |

File 4, Rung 5

Interrupt occurred due to low preset reached.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|-------|--------|
| 20 | -] [- | LD | interrupt occurred due to low preset reached C0/IL | 0 | |
| 134 | | RET | | | |

File 4, Rung 6

Signals the main program (file 2) to initiate a drilling sequence. The high-speed counter has already stopped the conveyor at the correct position using its high preset output pattern data (clear O:0/0). This occurred within microseconds of the high preset being reached (just prior to entering this high-speed counter interrupt subroutine). The drill sequence subroutine resets the drill sequence start bit and sets the conveyor drive bit (O:0/0) upon completion of the drilling sequence.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|-------|--------|
| 20 | -] [- | LD | interrupt occurred due to hi preset reached C0/IH | 0 | |
| 41 | -(L)- | SET | Drill Sequence Start B/32 | 0 | |

File 6, Rung 0

This section of ladder logic controls the up/down motion of the drill for the book drilling machine. When the conveyor positions the book under the drill, the DRILL SEQUENCE START bit is set. This rung uses that bit to begin the drilling operation. Because the bit is set for the entire drilling operation, the OSR is required to be able to turn off the forward signal so the drill can retract.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|------------------------------|-------|--------|
| 20 | -] [- | LD | Drill Sequence Start B/32 | 0 | |
| 29 | -OSR- | AND-OSR | Drill Subr OSR B/48 | 0 | |
| 41 | -(L)- | SET | Drill Forward O/3 | 0 | |

File 6, Rung 1

When the drill has drilled through the book, the body of the drill actuates the DRILL DEPTH limit switch. When this happens, the DRILL FORWARD signal is turned off and the DRILL RETRACT signal is turned on. The drill is also retracted automatically on power up if it is not actuating the DRILL HOME limit switch.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| 20 | -] [- | LD | Drill Depth LS I/4 | 0 | |
| 20 | -] [- | LD | 1'st Pass S1/15 | 0 | |
| 23 | -]/[- | ANI | Drill Home LS I/5 | 0 | |
| 14 | | ORB | | | |
| 42 | -(U)- | RST | Drill Forward O/3 | 0 | |
| 41 | -(L)- | SET | Drill Retract O/2 | 0 | |

File 6, Rung 2

When the drill is retracting (after drilling a hole), the body of the drill actuates the DRILL HOME limit switch. When this happens the DRILL RETRACT signal is turned off, the DRILL SEQUENCE START bit is turned off to indicate the drilling process is complete, and the conveyor is restarted.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|------------------------------|-------|--------|
| 20 | -] [- | LD | Drill Home LS I/5 | 0 | |
| 22 | -] [- | AND | Drill Retract O/2 | 0 | |
| 42 | -(U)- | RST | Drill Retract O/2 | 0 | |
| 42 | -(U)- | RST | Drill Sequence Start B/32 | 0 | |
| 41 | -(L)- | SET | Conveyor Start/Stop O/0 | 0 | |

File 7, Rung 0

Examines the number of 1/4 in. thousands that have accumulated over the life of the current drill bit. If the bit has drilled between 100,000-101,999 1/4 in. increments of paper, the "change drill" light illuminates steadily. When the value is between 102,000-103,999, the "change drill" light flashes at a 1.28 second rate. When the value reaches 105,000, the "change drill" light flashes and the "change drill now" light illuminates.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-----------------|-------------------|----------|--|--------------|--------|
| ---- | ----- | ----- | ----- | ----- | ----- |
| 10 | | MPS | | | |
| 66 | -GEQ- | AND-GEQ | 1/4 in. Thousands SRCA N11 SRCB | 0000H 100 | |
| 40 | -()- | OUT | 100,000 1/4 in. increments have occurred B/16 | 0 | |
| 11 | | MRD | | | |
| 66 | -GEQ- | AND-GEQ | 1/4 in. Thousands SRCA N11 SRCB | 0000H 102 | |
| 40 | -()- | OUT | 102,000 1/4 in. increments have occurred B/17 | 0 | |
| 11 | | MRD | | | |
| 66 ^① | -GEQ- | AND-GEQ | 1/4 in. Thousands SRCA N11 SRCB | 0000H 105 | |
| 40 | -()- | OUT | change drill bit NOW O/6 | 0 | |
| 12 | | MPP | | | |
| 20 | -] [- | LD | 100,000 1/4 in. increments have occurred B/16 | 0 | |
| 23 | -] / [- | ANI | 102,000 1/4 in. increments have occurred B/17 | 0 | |
| 20 | -] [- | LD | 100,000 1/4 in. increments have occurred B/16 | 0 | |
| 22 | -] [- | AND | 102,000 1/4 in. increments have occurred B/17 | 0 | |
| 22 | -] [- | AND | 1.28 second free running clock bit S4/7 | 0 | |
| 14 | | ORB | | | |
| 13 | | ANB | | | |
| 40 | -()- | OUT | change drill bit soon O/4 | 0 | |

① This branch accesses I/O only available with 32 I/O controllers. Therefore, do not include this branch if you are using a 16 I/O controller.

File 7, Rung 1

Resets the number of 1/4 in. increments and the 1/4 in. thousands when the "drill change reset" keyswitch is energized. This should occur following each drill bit change.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--------------------------------|----------------|--------|
| 20 | -] [- | LD | drill change reset I/8 | keyswitch 0 | |
| 85 | | CLR | 1/4 in. Thousands DEST N11 | 0000H | |
| 85 | | CLR | 1/4 in. increments DEST N10 | 0000H | |

File 7, Rung 2^①

Moves the single-digit BCD thumbwheel value into an internal integer register. This is done to properly align the four BCD input signals prior to executing the BCD to Integer instruction (FRD). The thumbwheel is used to allow the operator to enter the thickness of the paper that is to be drilled. The thickness is entered in 1/4 in. increments. This provides a range of 1/4 in. to 2.25 in.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| 10 | | MPS | | | |
| 22 | -] [- | AND | BCD bit 0 I/11 | 0 | |
| 40 | - () - | OUT | FRD bit 0 N14/0 | 0 | |
| 11 | | MRD | | | |
| 22 | -] [- | AND | BCD bit 1 I/12 | 0 | |
| 40 | - () - | OUT | FRD bit 1 N14/1 | 0 | |
| 11 | | MRD | | | |
| 22 | -] [- | AND | BCD bit 2 I/13 | 0 | |
| 40 | - () - | OUT | FRD bit 2 N14/2 | 0 | |
| 12 | | MPP | | | |
| 22 | -] [- | AND | BCD bit 3 I/14 | 0 | |
| 40 | - () - | OUT | FRD bit 3 N14/3 | 0 | |

① This rung accesses I/O only available with 32 I/O controllers. Therefore, do not include this rung if you are using a 16 I/O controller.

File 7, Rung 3

Converts the BCD thumbwheel value from BCD to integer. This is done because the controller operates upon integer values. This rung also "debounces" the thumbwheel to ensure that the conversion only occurs on valid BCD values. Note that invalid BCD values can occur while the operator is changing the BCD thumbwheel. This is due to input filter propagation delay differences between the 4 input circuits that provide the BCD input value.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|----------------|--------|
| 10 | | MPS | | | |
| 23 | -]/[- | ANI | 1'st pass bit S1/15 | 0 | |
| 51 | -EQU- | AND-EQU | previous scan's BCD input value SRCA N13 SRCB N14 | 0000H 0000H | |
| 101 | | FRD | debounced BCD value SRC N14 DEST N12 | 0000H 0000H | |
| 22 | -] [- | AND | Math Overflow Bit S0/1 | 0 | |
| 42 | -(U)- | RST | Math Error Bit S5/0 | 0 | |
| 12 | | MPP | | | |
| 106 | | MOV | this scan's BCD input value SRC N14 DEST N13 | 0000H 0000H | |

File 7, Rung 4

Ensures that the operator cannot select a paper thickness of 0. If this were allowed, the drill bit life calculation could be defeated resulting in poor quality holes due to a dull drill bit. Therefore, the minimum paper thickness used to calculate drill bit wear is 1/4 in.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|------------|--------|
| 50 | -EQU- | LD-EQU | debounced BCD value SRCA N12 SRCB | 0000H 0 | |
| 106 | | MOV | debounced BCD value SRC DEST N12 | 1 0000H | |

File 7, Rung 5

Keeps a running total of how many inches of paper have been drilled with the current drill bit. Every time a hole is drilled, adds the thickness (in 1/4 ins) to the running total (kept in 1/4 ins). The OSR is necessary because the ADD executes every time the rung is true, and the drill body would actuate the DRILL DEPTH limit switch for more than 1 program scan. Integer N7:12 is the integer-converted value of the BCD thumbwheel on inputs I:0/11 - I:0/14.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|-------------------------|--------|
| 20 | -] [- | LD | Drill Depth LS I/4 | 0 | |
| 29 | -OSR- | AND-OSR | Tool Wear OSR 1 B/24 | 0 | |
| 80 | | ADD | SRCA N12 1/4 in. increments SRCB N10 1/4 in. increments DEST N10 | 0000H 0000H 0000H | |

File 7, Rung 6

When the number of 1/4 in. increments surpasses 1000, finds out how many increments are past 1000 and stores in N7:20. Adds 1 to the total of '1000 1/4 in.' increments, and re-initializes the 1/4 in. increments accumulator to how many increments were beyond 1000.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|------------------------|--------|
| 65 | - GEQ - | LD-GEQ | 1/4 in. increments SRCA N10 SRCB | 0000H 1000 | |
| 81 | | SUB | 1/4 in. increments SRCA N10 SRCB DEST N20 | 0000H 1000 0000H | |
| 80 | | ADD | SRCA 1/4 in. Thousands SRCB N11 1/4 in. Thousands DEST N11 | 1 0000H 0000H | |
| 106 | | MOV | SRC N20 1/4 in. increments DEST N10 | 0000H 0000H | |

Time-Driven Sequencer Application Example

The following application example illustrates the use of the TON and SQO instructions in a traffic signal at an intersection. The timing requirements are:

- Red light – 30 seconds
- Yellow light – 15 seconds
- Green light – 60 seconds

The timer, when it reaches its preset, steps the sequencer that in turn controls which traffic signal is illuminated. For a detailed explanation of:

- LD, LDI, and TON instructions, see chapter 8.
- SQO and SQC instructions, see chapter 13.

Time Driven Sequencer Ladder Program

Rung 2:0

The function of this rung is called a regenerative timer. Every time the timer reaches its preset, the DONE bit is set for one scan--this causes this rung to become FALSE for one scan and resets the timer. On the following scan, when this rung becomes TRUE again, the timer begins timing.

| | |
|---|--|
| <pre> Timer Enable T4:0 -----] / [----- DN </pre> | <pre> Timer +TON-----+ +TIMER ON DELAY +- (EN)- Timer T4:0+- (DN) Time Base 0.01 Preset 1 Accum 0 +-----+ </pre> |
|---|--|

Rung 2:1

Controls the RED, GREEN, and YELLOW lights wired to outputs 0:0/0 - 0:0/2 and controls how long the regenerative timer times between each step. When this rung goes from false-to-true (by the timer reaching its preset), the first sequencer changes which traffic light is illuminated and the second sequencer changes the preset of the timer to determine how long this next light is illuminated.

| | |
|---|--|
| <pre> T4:0 --] [----- DN </pre> | <pre> RED, GREEN, and YELLOW lights +SQO-----+ +SEQUENCER OUTPUT +- (EN)-+ File #N7:0+- (DN) Mask 0007+ Dest 0:0.0 Control R6:0 Length 3 Position 0 +-----+ Timer Presets for each lights +SQO-----+ +SEQUENCER OUTPUT +- (EN)-+ File #N7:5+- (DN) Mask FFFF Dest T4:0.PRE Control R6:1 Length 3 Position 0 +-----+ </pre> |
|---|--|

Rung 2.2

| |
|--------------------------------------|
| <pre> -----+END+----- </pre> |
|--------------------------------------|

Data Files

| Address | 15 | Data | | | 0 |
|---------|------|------|------|------|------|
| N7:0 | 0000 | 0000 | 0000 | 0000 | 0000 |
| N7:1 | 0000 | 0000 | 0000 | 0000 | 0100 |
| N7:2 | 0000 | 0000 | 0000 | 0000 | 0010 |
| N7:3 | 0000 | 0000 | 0000 | 0000 | 0001 |

Data Table

| Address | Data | (Radix=Decimal) | | | | | | | |
|---------|------|-----------------|---|---|---|---|------|------|------|
| N7:0 | 0 | 4 | 2 | 1 | 0 | 0 | 6000 | 1500 | 3000 |

Time Driven Sequencer Instruction List Program

File 2, Rung 0

The function of this rung is called a regenerative timer. Every time the timer reaches its preset, the DONE bit is set for one scan--this causes this rung to become FALSE for one scan and resets the timer. On the following scan, when this rung becomes TRUE again, the timer begins timing.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME | ADDRESS | VALUE | FORCES |
|----------|----------------|----------|----------------|---------|-------|--------|
| 21 | - /[- | LDI | Timer | Enable | ----- | ----- |
| | | | | T0/DN | 0 | |
| 0 | | TON | Timer | TIMR T0 | | |
| | | | | BASE | 0.01 | |
| | | | | PRE | 0001H | |
| | | | | ACC | 0000H | |

File 2, Rung 1

Controls the RED, GREEN, and YELLOW lights wired to outputs O:0/0 - O:0/2 and controls how long the regenerative timer times between each step. When this rung goes from false-to-true (by the timer reaching its preset), the first sequencer changes which traffic light is illuminated and the second sequencer changes the preset of the timer to determine how long this next light is illuminated.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME | ADDRESS | VALUE | FORCES |
|----------|----------------|----------|------------------------------|---------|-------|--------|
| 20 | - [- | LD | Step | SQ0 | ----- | ----- |
| | | | | T0/DN | 0 | |
| 152 | | SQ0 | RED, GREEN and YELLOW lights | | | |
| | | | FILE #N0 | | | |
| | | | MASK | O0 | 0007H | |
| | | | DEST | R0 | | |
| | | | CTRL | R0 | | |
| | | | LEN | | 0003H | |
| | | | POS | | 0000H | |
| 152 | | SQ0 | Timer presets for each light | | | |
| | | | FILE #N5 | | | |
| | | | MASK | | FFFFH | |
| | | | DEST | T0.PRE | | |
| | | | CTRL | R1 | | |
| | | | LEN | | 0003H | |
| | | | POS | | 0000H | |

Event-Driven Sequencer Application Example

The following application example illustrates how the FD (found) bit on an SQC instruction can be used to advance an SQO to the next step (position). This application program is used when a specific order of events is required to occur repeatedly. By using this combination, you can eliminate using the XIO, XIC, and other instructions. For a detailed explanation of:

- LD, LDI, and RES instructions, see chapter 8.
- SQO and SQC instructions, see chapter 13.

Event Driven Sequencer Ladder Program

Rung 2:0

Ensures that the SQO always resets to step (position) 1 each RRUN mode entry. (This rung actually resets the control register's position and EN enable bit to 0. Due to this, the following rung sees a false to true transition and asserts step (position) 1 on the first scan.)

Eliminate this rung for retentive operation.

```

S:1
--] [-----R6:0
      15      (RES)-----

```

Rung 2:1

The SQC instruction and SQO instruction share the same Control Register. This is acceptable due to the careful planning of the rung state condition. You could cascade (branch) many more SQO instructions below the SQO if you desired, all using the same Control Register (R6:0 in this case). Notice that we are only comparing Inputs 0-3 and are only asserting Outputs 0-3 (per our Mask value).

```

R6:0
--]/[-----+SQ-----+
      FD      SEQUENCER COMPARE +- (EN) +-
              File      #N7:0+- (DN)
              Mask      000F+- (FD)
              Source     I:0.0
              Control    R6:0
              Length     9
              Position   2
              +-----+
R6:0 +SQO-----+
+--]/[--+SEQUENCER OUTPUT +- (EN) +-
      FD  File      #N7:10+- (DN)
          Mask      000F
          Dest      0:0.0
          Control   R6:0
          Length    9
          Position  2
          +-----+

```

Rung 2.2

```

-----+END+-----

```

The following displays the FILE DATA for both sequencers. The SQC compare data starts at N7:0 and ends at N7:9. While the SQO output data starts at N7:10 and ends at N7:19. Please note that step 0 of the SQO is never active. The reset rung combined with the rung logic of sequencers guarantees that the sequencers always start at step 1. Both sequencers also "roll over" to step 1. "Roll Over" to step 1 is integral to all sequencer instructions.

SQC Compare Data

| Addresses | Data (Radix=Decimal) | | | | | | | | | |
|-----------|----------------------|---|---|---|---|---|---|---|---|---|
| N7:0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| N7:10 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Event Driven Sequencer Instruction List Program

File 2, Rung 0

Ensures that the SQO always resets to step (position) 1 each RRUN mode entry. (This rung actually resets the control register's position and EN enable bit to 0. Due to this the following rung sees a false to true transition and asserts step (position) 1 on the first scan.)

Eliminate this rung for retentive operation.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| ---- | ----- | ----- | ----- | ----- | ----- |
| 20 | -] [- | LD | S1/15 | 0 | |
| 7 | | RES | R0 | | |

File 2, Rung 1

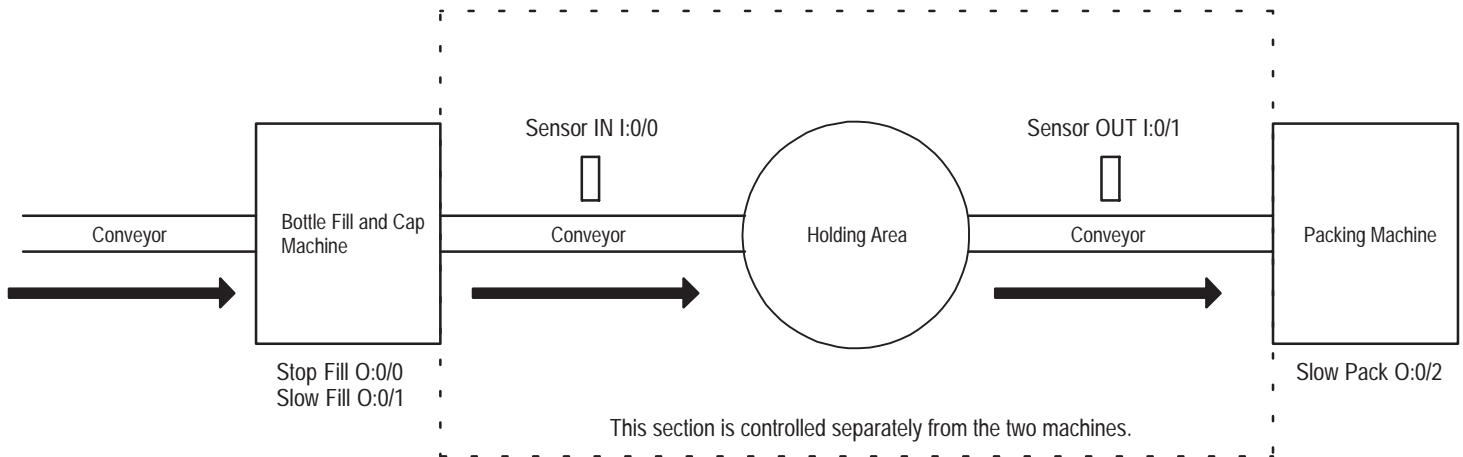
The SQC instruction and SQO instruction share the same Control Register. This is acceptable due to the careful planning of the rung state condition. You could cascade (branch) many more SQO instructions below the SQO if you desired, all using the same Control Register (R6:0 in this case). Notice that we are only comparing Inputs 0-3 and are only asserting Outputs 0-3 (per our Mask value).

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|-------------------------|--------|
| ---- | ----- | ----- | ----- | ----- | ----- |
| 20 | -] / [- | LDI | R0/FD | 0 | |
| 153 | | SQC | FILE #N0 MASK SRC IO CTRL R0 LEN POS | 000FH 9 0000H | |
| 23 | -] / [- | ANI | R0/FD | 0 | |
| 152 | | SQO | FILE #N10 MASK DEST O0 CTRL R0 LEN POS | 000FH 9 0000H | |

Bottle Line Example

The following application example illustrates how the controller high-speed counter is configured for an up/down counter. For a detailed explanation of:

- LD, SET, RST, and OUT instructions, see chapter 8.
- GRT, LES, and GEQ instruction, see chapter 9.
- HSC and HSL instructions, see chapter 14.



Bottle Line Operation Overview

The controller on the conveyor, within the specified area above, regulates the speeds of the bottle fill and packing machines. Each machine is connected to a separate controller that communicates with the conveyor controller. The following ladder program is for the conveyor controller.

A conveyor feeds filled bottles past a proximity sensor (IN) to a holding area. The proximity sensor is wired to the I/0 terminal (up count) of the conveyor controller. The bottles are then sent on another conveyor past a proximity switch (OUT) to the packing machine. This proximity switch is wired to the I/1 terminal (down count) on the same controller.

Bottle Line Ladder Program

```

Rung 2:0
Loads the high-speed counter with the following parameters:
N7:0 - 0001h Output Mask - Effect only O:0/0
N7:1 - 0001h Output Pattern for High Preset - Energize O:0/0 upon high preset
N7:2 - 350d High Preset - Maximum numbers of bottles for the holding area
N7:3 - 0000h Output Pattern for Low Preset - not used
N7:4 - 0d Low Preset - not used

```

| | |
|---------------|-----------------|
| First Pass | |
| Bit | |
| S:1 | +HSL-----+ |
| -----] [----- | +HSC LOAD-----+ |
| 15 | Counter C5:0 |
| | Source N7:0 |
| | Length 5 |
| | +-----+ |

Rung 2:1

Starts up the high-speed counter with the above parameters. Each time the rung is evaluated, the hardware accumulator is written to C5:0.ACC.

| | |
|----------------------------|---------------|
| +HSC-----+ | |
| +HIGH SPEED COUNTER+-(CU)- | |
| Type | Up/Down+-(CD) |
| Counter | C5:0+-(DN) |
| Preset | 350 |
| Accum | 0 |
| +-----+ | |

Rung 2:2

Packing machine running too fast for the filling machine. Slow down the packing machine to allow the filler to catch up.

| | | |
|------------|----------|-----------|
| +LES-----+ | | Slow Pack |
| +LESS THAN | | O:0 |
| Source A | C5:0.ACC | (L)----- |
| | 0 | 2 |
| Source B | 100 | |
| +-----+ | | |

Rung 2:3

If the packer was slowed down to allow the filler to catch up, wait until the holding area is approximately 2/3 full before allowing the packer to run at full speed again.

| | | | |
|---------------|----------|---------------|-----------|
| +GRT-----+ | | Slow Pack | Slow Pack |
| +GREATER THAN | | O:0 | O:0 |
| Source A | C5:0.ACC | -----] [----- | (U)----- |
| | 0 | 2 | 2 |
| Source B | 200 | | |
| +-----+ | | | |

Rung 2:4

Filling machine running too fast for the packing machine. Slow down the filling machine to allow the packer to catch up.

| | | | |
|---------------|----------|---------------|-----------|
| +GRT-----+ | | Slow Fill | Slow Fill |
| +GREATER THAN | | O:0 | O:0 |
| Source A | C5:0.ACC | -----] [----- | (L)----- |
| | 0 | 1 | 1 |
| Source B | 250 | | |
| +-----+ | | | |

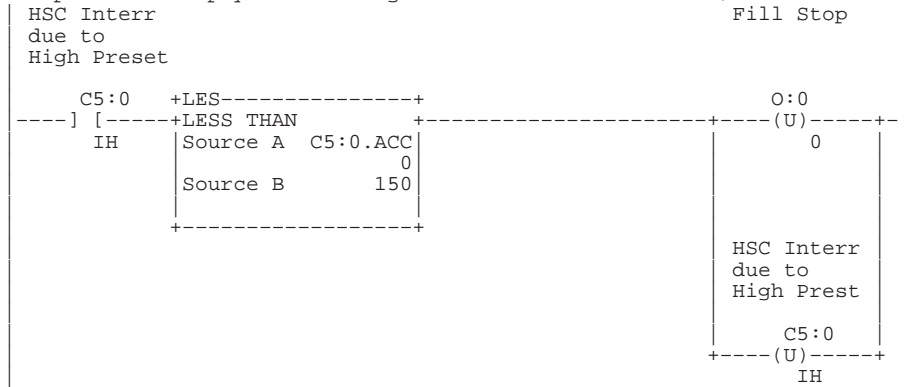
Rung 2:5

If the filler was slowed down to allow the packer to catch up, wait until the holding area is approximately 1/3 full before allowing the filler to run at full speed again.

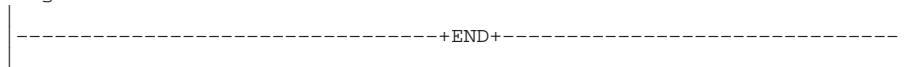
| | | | |
|------------|----------|---------------|-----------|
| +LES-----+ | | Slow Fill | Slow Fill |
| +LESS THAN | | O:0 | O:0 |
| Source A | C5:0.ACC | -----] [----- | (U)----- |
| | 0 | 1 | 1 |
| Source B | 150 | | |
| +-----+ | | | |

Rung 2:6

If the high-speed counter reached its high preset of 350 (indicates that the holding area reached maximum capacity), it would energize O:0/0, shutting down the filling operation. Before re-starting the filler, allow the packer to empty the holding area until it is about 1/3 full.



Rung 2:7



Data Table

| Addresses | Data | (Radix=Decimal) |
|-----------|------|-----------------|
| N7:0 | 1 1 | 350 0 0 |

Bottle Line Instruction List Program

File 2, Rung 0

Loads the high-speed counter with the following parameters:

N7:0 - 0001h Output Mask - Effect only O:0/0
 N7:1 - 0001h Output Pattern for High Preset - Energize O:0/0 upon high preset
 N7:2 - 350d High Preset - Maximum numbers of bottles for the holding area
 N7:3 - 0000h Output Pattern for Low Preset - not used
 N7:4 - 0d Low Preset - not used

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|-------|--------|
| 20 | -] [- | LD | First Pass Bit S1/15 | 0 | |
| 171 | | HSL | CNTR C0 SRC N0 LEN | 5 | |

File 2, Rung 1

Starts up the high-speed counter with the above parameters. Each time the rung is evaluated, the hardware accumulator is written to C5:0.ACC.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|-------------------------------|---------------------------|--------|
| 170 | | HSC | TYPE CNTR C0 PRE ACC | Up/Down 015EH 0000H | |

File 2, Rung 2

Packing machine running too fast for the filling machine. Slow down the packing machine to allow the filler to catch up.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|--------------|--------|
| 56 | - LES - | LD-LES | SRCA C0.ACC SRCB | 0000H 100 | |
| 41 | - (L) - | SET | Slow Pack O/2 | 0 | |

File 2, Rung 3

If the packer was slowed down to allow the filler to catch up, wait until the holding area is approximately 2/3 full before allowing the packer to run at full speed again.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|--------------|--------|
| 62 | - GRT - | LD-GRT | SRCA C0.ACC SRCB | 0000H 200 | |
| 22 | -] [- | AND | Slow Pack O/2 | 0 | |
| 42 | - (U) - | RST | Slow Pack O/2 | 0 | |

File 2, Rung 4

Filling machine running too fast for the packing machine. Slow down the filling machine to allow the packer to catch up.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|--------------|--------|
| 62 | -GRT- | LD-GRT | SRCA C0.ACC SRCB | 0000H 250 | ----- |
| 41 | -(L)- | SET | Slow Fill O/1 | 0 | |

File 2, Rung 5

If the filler was slowed down to allow the packer to catch up, wait until the holding area is approximately 1/3 full before allowing the filler to run at full speed again.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|--------------|--------|
| 56 | -LES- | LD-LES | SRCA C0.ACC SRCB | 0000H 150 | ----- |
| 22 | -] [- | AND | Slow Fill O/1 | 0 | |
| 42 | -(U)- | RST | Slow Fill O/1 | 0 | |

File 2, Rung 6

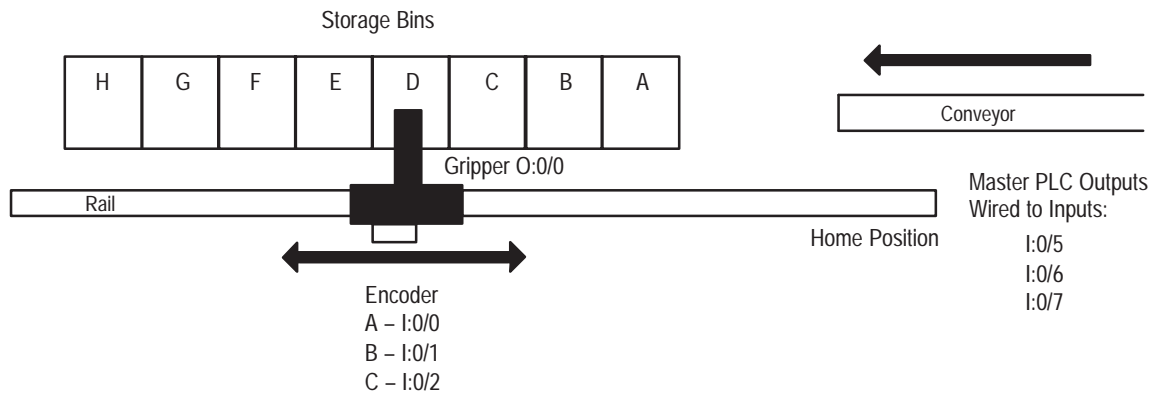
If the high-speed counter reached its high preset of 350 (indicates that the holding area reached maximum capacity), it would energize O:0/0, shutting down the filling operation. Before re-starting the filler, allow the packer to empty the holding area until it is about 1/3 full.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------------------|--------------|--------|
| 20 | -] [- | LD | HSC Interr due to High Prest C0/IH | 0 | ----- |
| 57 | -LES- | AND-LES | SRCA C0.ACC SRCB | 0000H 150 | |
| 42 | -(U)- | RST | Fill Stop O/0 | 0 | |
| 42 | -(U)- | RST | HSC Interr due to High Prest C0/IH | 0 | |

Pick and Place Machine Example

The following application example illustrates how the controller high-speed counter is configured for the up and down counter using an encoder with reset and hold. For a detailed explanation of:

- LD, LDI, OUT, RES, SET, RST, and TON instructions, see chapter 8.
- GRT and NEQ instructions, see chapter 9.
- MOV instruction, see chapter 11.
- HSC and HSL instructions, see chapter 14.



Pick and Place Machine Operation Overview

A pick and place machine takes parts from a conveyor and drops them into the appropriate bins. When the pick and place head is positioned over the conveyor with a gripped part, the master PLC communicates to the controller controlling the gripper which bin to drop the part into. This information is communicated by energizing three outputs that are wired to the controller's inputs.

Once the controller has this information, it grabs the part and moves down the rail. When the gripper reaches the appropriate bin, it opens and the part falls into the bin. The gripper then returns to the conveyor to retrieve another part.

The position of the pick and place head is read by the controller via a 1000 line quadrature encoder wired to the controller's high-speed counter inputs. When the gripper is in the home position, the Z pulse from the encoder resets the high-speed counter. The number of pulses the head needs to travel to reach each bin location is stored in a data table starting at address N10 and ending at N17. The controller uses indexed addressing to locate the correct encoder count from the data table and to load the information into the high preset of the high-speed counter.

Pick and Place Machine Ladder Program

Rung 2:0

The following three rungs take information from the other programmable controller and load it into the INDEX REGISTER. This is used to select the proper bin location from the table starting at N7:10.



Rung 2:1

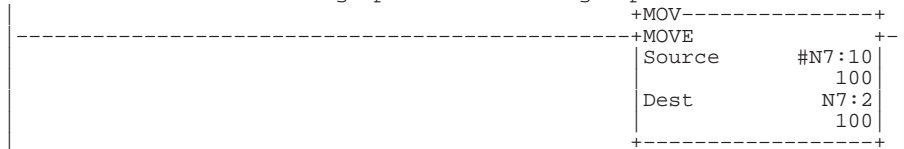


Rung 2:2



Rung 2:3

Indexes into the table of bin locations and places the correct number of encoder counts into the high preset of the high-speed counter.



Rung 2:4

Loads the high-speed counter with the following parameters:

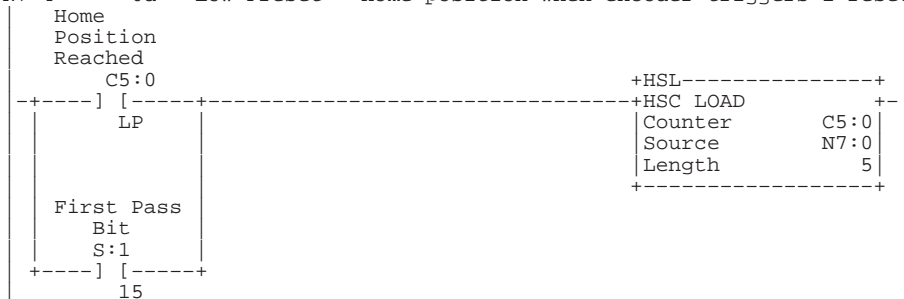
N7:0 - 0001h - Output Mask - high-speed counter control only 0:0/0 (gripper)

N7:1 - 0000h - Output Pattern for High Preset - turn OFF gripper (release part)

N7:2 - 100d - High Preset - loaded from table in the rung above

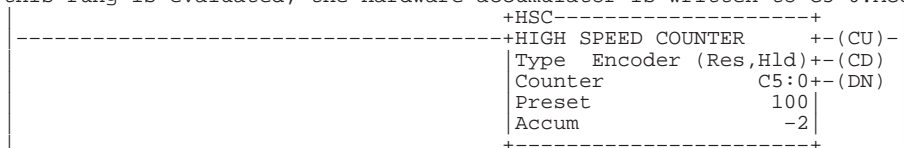
N7:3 - 0001h - Output Pattern for Low Preset - turn ON gripper (grab part)

N7:4 - 0d - Low Preset - home position when encoder triggers Z-reset



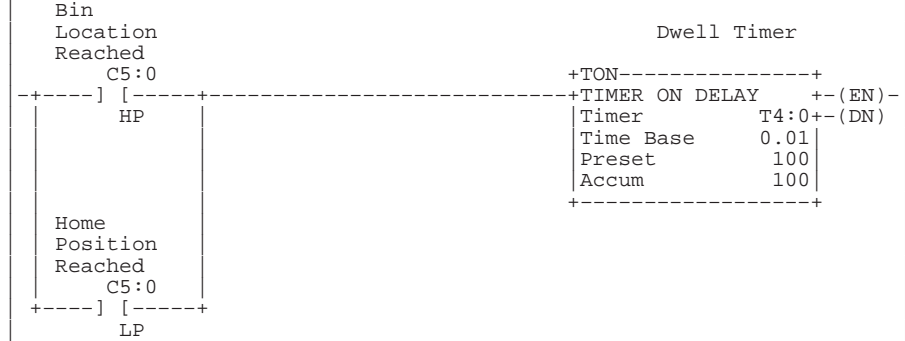
Rung 2:5

Starts up the high-speed counter with the above parameters. Each time this rung is evaluated, the hardware accumulator is written to C5:0.ACC.



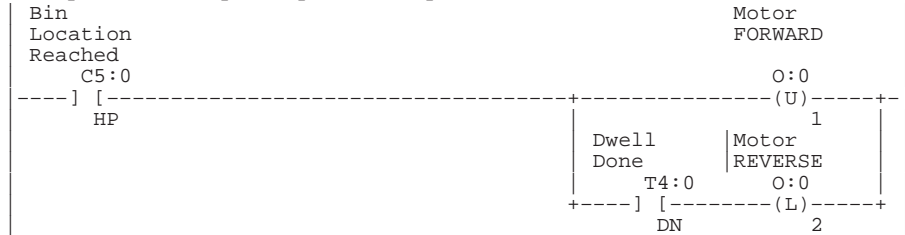
Rung 2:6

When the pick-and-place head reaches either its home position to pick up a part or its destination bin to drop off a part, start up a dwell timer. The purpose of this is to keep the head stationary long enough for the gripper to either grab or release the part.



Rung 2:7

When the pick-and-place head is positioned over the proper bin, turn off the forward motor. At the same time, the high-speed counter tells the gripper to release the part and start the dwell timer. After the dwell time has expired, start up the reverse motor to send the head back to its home position to pick up another part.

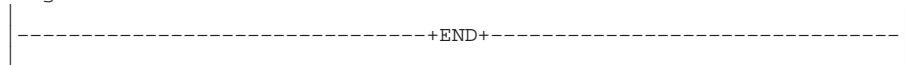


Rung 2:8

When the pick-and-place head is positioned at its home position, turn off the reverse motor. At the same time, the high-speed counter tells the gripper to grab the next part and start the dwell timer. After the dwell time has expired, start up the forward motor to send the head out to its drop off bin.



Rung 2:9



Data Table

| Addresses | Data | (Radix=Decimal) |
|-----------|-------------------------------------|-----------------|
| N7:0 | 1 0 100 1 0 0 0 0 0 0 | 0 |
| N7:10 | 100 200 300 400 500 600 700 800 0 0 | 0 |

Pick and Place Machine Instruction List Program

File 2, Rung 0

The following three rungs take information from the other programmable controller and load it into the INDEX REGISTER. This is used to select the proper bin location from the table starting at N7:10.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|----------------------------|-------|--------|
| 20 | -] [- | LD | Output from barcode I/5 | 0 | |
| 40 | - () - | OUT | Index Reg S24/0 | 0 | |

File 2, Rung 1

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|----------------------------|-------|--------|
| 20 | -] [- | LD | Output from barcode I/6 | 0 | |
| 40 | - () - | OUT | Index Reg S24/1 | 0 | |

File 2, Rung 2

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|----------------------------|-------|--------|
| 20 | -] [- | LD | Output from barcode I/7 | 0 | |
| 40 | - () - | OUT | Index Reg S24/2 | 0 | |

File 2, Rung 3

Indexes into the table of bin locations and places the correct number of encoder counts into the high preset of the high-speed counter.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---------------------------|------------|--------|
| 106 | | MOV | SRC #N10 DEST N2 | 100 100 | |

File 2, Rung 4

Loads the high-speed counter with the following parameters:
 N7:0 - 0001h - Output Mask - high-speed counter control only 0:0/0
 (gripper)
 N7:1 - 0000h - Output Pattern for High Preset - turn OFF gripper (release
 part)
 N7:2 - 100d - High Preset - loaded from table in the rung above
 N7:3 - 0001h - Output Pattern for Low Preset - turn ON gripper (grab part)
 N7:4 - 0d - Low Preset - home position when encoder triggers Z-reset

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|-----------------------|---------|-------|--------|
| | | | NAME | ADDRESS | | |
| 20 | -] [- | LD | Home Position Reached | C0/LP | 0 | |
| 24 | _] [_ | OR | First Pass Bit | S1/15 | 0 | |
| 171 | | HSL | CNTR | C0 | | |
| | | | SRC | N0 | | |
| | | | LEN | | 5 | |

File 2, Rung 5

Starts up the high-speed counter with the above parameters. Each time this rung is evaluated the hardware accumulator is written to C5:0.ACC.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|-----------|---------|-------------------|--------|
| | | | NAME | ADDRESS | | |
| 170 | | HSC | TYPE | | Encoder (Res,Hld) | |
| | | | CNTR | C0 | | |
| | | | PRE | | 100 | |
| | | | ACC | | -2 | |

File 2, Rung 6

When the pick-and-place head reaches either its home position to pick up a part or its destination bin to drop off a part, start up a dwell timer. The purpose of this is to keep the head stationary long enough for the gripper to either grab or release the part.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|-----------------------|---------|-------|--------|
| | | | NAME | ADDRESS | | |
| 20 | -] [- | LD | Bin Location Reached | C0/HP | 0 | |
| 24 | _] [_ | OR | Home Position Reached | C0/LP | 0 | |
| 0 | | TON | Dwell Timr | TIMR T0 | | |
| | | | BASE | | 0.01 | |
| | | | PRE | | 100 | |
| | | | ACC | | 100 | |

File 2, Rung 7

When the pick-and-place head is positioned over the proper bin, turn off the forward motor. At the same time, the high-speed counter tells the gripper to release the part and start the dwell timer. After the dwell time has expired, start up the reverse motor to send the head back to its home position to pick up another part.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|-------------------------------|-------|--------|
| 20 | -] [- | LD | Bin Location Reached C0/HP | 0 | |
| 42 | -(U)- | RST | Motor FORWARD O/1 | 0 | |
| 22 | -] [- | AND | Dwell Done T0/DN | 0 | |
| 41 | -(L)- | SET | Motor REVERSE O/2 | 0 | |

File 2, Rung 8

When the pick-and-place head is positioned at its home position, turn off the reverse motor. At the same time, the high-speed counter tells the gripper to grab the next part and start the dwell timer. After the dwell time has expired, start up the forward motor to send the head out to its drop off bin.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--------------------------------|-------|--------|
| 20 | -] [- | LD | Home Position Reached C0/LP | 0 | |
| 42 | -(U)- | RST | Motor REVERSE O/2 | 0 | |
| 22 | -] [- | AND | Dwell Done T0/DN | 0 | |
| 41 | -(L)- | SET | Motor FORWARD O/1 | 0 | |

RPM Calculation Application Example

The following application example illustrates how to calculate the frequency and RPM of a device (such as an encoder) connected to a high-speed counter. *The calculated values are only valid when counting up.* For a detailed explanation of:

- LD, LDI, CTU and TON instructions, see chapter 8.
- LES instruction, see chapter 9.
- CLR, MUL, DIV, DDV, ADD, and SUB instructions, see chapter 10.
- MOV instruction, see chapter 11.

RPM Calculation Operation Overview

This is done by manipulating the number of counts that have occurred in the high-speed counter accumulator (C0.ACC) over time. To determine this you must provide the following application specific information:

- N2 – Counts per Revolution. (i.e., the number of encoder pulses per revolution i.e., the number of pulses until reset). This value is entered in whole counts. For example, you would enter the value 1000 into N2 for a 1000 count A/B/Z encoder.
- T0.PRE– The Rate Measurement Period (i.e., the amount of time in which to sample the accumulation of counts). This value is entered in .01 second intervals. For example, enter the value 10 into T0.PRE for a .1 second rate measurement period. For an accurate frequency and RPM calculation to occur, the value entered must divide evenly into 100. For example valid=20,10,5,4,2,1 and invalid=11,9,8,7,6,3.

Once you have entered these two values the following information is provided:

- N1 – Counts per last Rate Measurement Period. This value is updated each end of Rate Measurement Period with the number of counts that have elapsed. Use this value if your application requires high-speed calculations such as velocity.
- N4 – Frequency. This value is updated once per second with the number of pulses that occurred in the last second. This value (frequency) is calculated:

$$\text{Frequency (Hz)} = \frac{\# \text{ pulses}}{1 \text{ second}}$$

- N5 – RPM. This value is calculated once per second using the frequency value N4 together with the counts per revolution value N2. For example, if N4 contained the value 2000 (indicates 2000 Hz) and you had specified a 1000 count encoder in N2, the RPM calculation for N5 would be 120. This equates to 2 encoder revolutions per second. Refer to the calculation below:

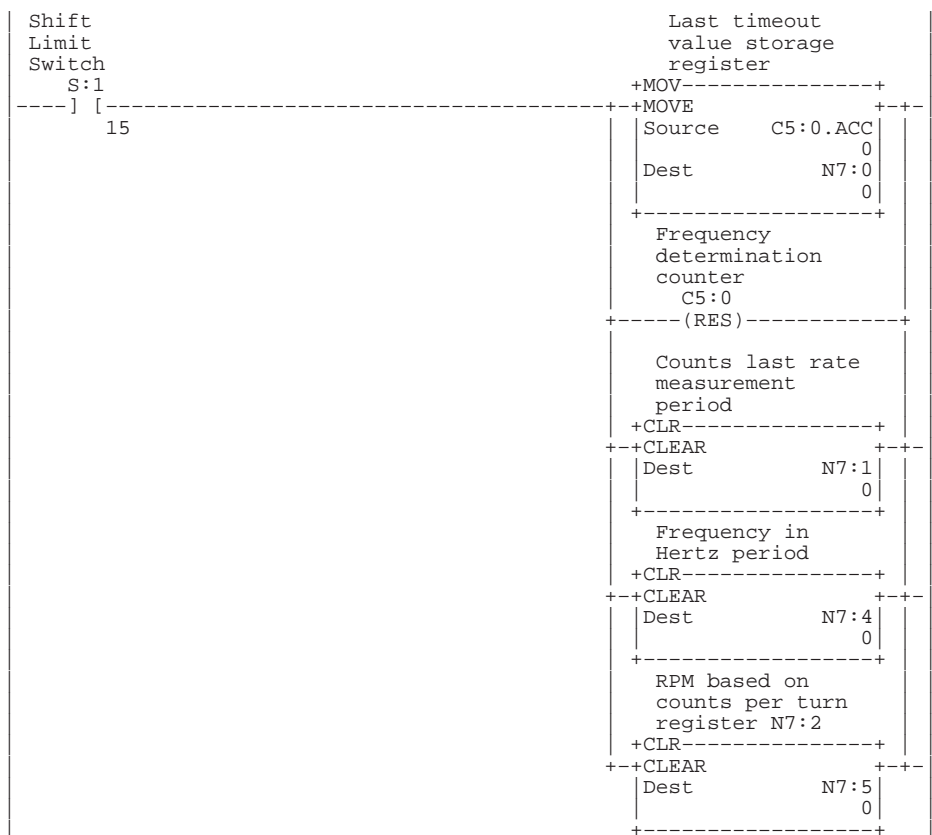
$$\text{RPM} = \frac{\# \text{ pulses}}{1 \text{ second}} \times \frac{1 \text{ revolution}}{\# \text{ pulses}} \times \frac{60 \text{ seconds}}{1 \text{ minute}}$$

$$120 \text{ RPM} = \frac{2000 \text{ pulses}}{1 \text{ second}} \times \frac{1 \text{ revolution}}{1000 \text{ pulses}} \times \frac{60 \text{ seconds}}{1 \text{ minute}}$$

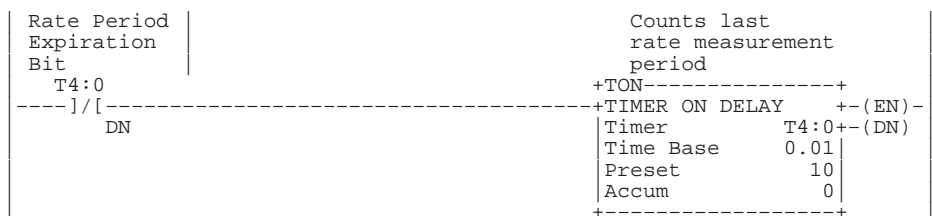
To maintain validity, you must ensure that you cannot accumulate more pulses per rate period than counts per revolution. For example, if you have selected a 1000 pulse encoder, you cannot have more than 999 counts occur in any 1 rate measurement period. If you determine that you exceed this rule, simply lower your Rate Measurement Period TO.PRE.

RPM Calculation Ladder Program

Rung 2:0
Ensures that the measurement value is initialized each RRUN mode entry.



Rung 2:1
Sets the rate measurement period. In this case, we are calculating a new rate value once every 100ms. Value N7:1 is updated once every 100ms with the number of counts that have occurred in the last 100ms period. Note that the preset value must divide evenly into 100 in order to accurately determine frequency and RPM (determined later in this program).



Appendix E
Application Example Programs

Rung 2:2

Calculates and stores the number of counts that have occurred since the last time that it was executed as true in N7:1 (last time=last rate measurement timer (T4:0) expiration). The LES instruction allows for 10 counts of backlash to occur (you can adjust as needed). The add instruction is configured for a 1000 count encoder using N7:2. (Change this register to match the number of counts generated each Z reset.)

| Rate Period Expiration Bit | | Counts last rate measurement period | | Counts last rate measurement period |
|-------------------------------|-----------------------------|--|---|--|
| T4:0 | | | +SUB-----+ | |
| --] [----- | | | +SUBTRACT | |
| DN | | | Source A C5:0.ACC | |
| | | | 0 | |
| | | | Source B N7:0 | |
| | | | 0 | |
| | | | Dest N7:1 | |
| | | | 0 | |
| | | | +-----+ | |
| | If negative math flag | Counts last rate measurement period | +ADD-----+ | Counts last rate measurement period |
| S:0 | +LES-----+ | | +ADD-----+ | |
| --] [----- | +LESS THAN | | +ADD-----+ | |
| 3 | Source A N7:1 | | Source A N7:2 | |
| | 0 | | 1000 | |
| | Source B -10 | | Source B N7:1 | |
| | | | 0 | |
| | | | Dest N7:1 | |
| | | | 0 | |
| | | | +-----+ | |
| | | | Last timeout value storage register | |
| | | | +MOV-----+ | |
| | | | +MOVE | |
| | | | Source C5:0.ACC | |
| | | | 0 | |
| | | | Dest N7:0 | |
| | | | 0 | |
| | | | +-----+ | |
| | | | Determine 1 second count. ie: # of rate periods | |
| | | | +DIV-----+ | |
| | | | +DIVIDE | |
| | | | Source A 100 | |
| | | | Source B T4:0.PRE | |
| | | | 10 | |
| | | | Dest C5:1.PRE | |
| | | | 10 | |
| | | | +-----+ | |
| | | | Frequency determination counter | |
| | | | +CTU-----+ | |
| | | | +COUNT UP | +- (CU) -+ |
| | | | Counter C5:1 | +- (DN) -+ |
| | | | Preset 10 | |
| | | | Accum 0 | |
| | | | +-----+ | |
| | | | Frequency calculation register | |
| | | | +ADD-----+ | |
| | | | +ADD | |
| | | | Source A N7:1 | |
| | | | 0 | |
| | | | Source B N7:3 | |
| | | | 0 | |
| | | | Dest N7:3 | |
| | | | 0 | |
| | | | +-----+ | |

```

1 second      Frequency
has now       in Hertz
elapsed
C5:1         +MOV-----+
+---] [-----+-----+-----+
DN           +MOVE
            |Source      N7:3
            |           0
            |Dest       N7:4
            |           0
            +-----+
            |Frequency
            |calculation
            |register
            +CLR-----+
            +CLEAR-----+
            |Dest       N7:3
            |           0
            +-----+
            |Frequency
            |determination
            |counter
            |C5:1
            |(RES)-----+
            |Temporary reg.
            |(math reg is real
            |destination
            +MUL-----+
            +MULTIPLY-----+
            |Source A   N7:4
            |           0
            |Source B   60
            |Dest      N7:6
            |           0
            +-----+
            |RPM based on
            |counts per turn
            |register N7:2
            +DDV-----+
            +DOUBLE DIVIDE-----+
            |Source    N7:2
            |          1000
            |Dest     N7:5
            |           0
            +-----+
            |Math overflow
            |error bit
            |S:5
            +(U)-----+
            |           0

```

Rung 2:3

```

+HSC-----+
+HIGH SPEED COUNTER+- (CU)-
|Type   Up (Res,Hld)+- (CD)
|Counter C5:0+- (DN)
|High Preset 1000
|Accum      0
+-----+

```

Rung 2:4

```

+-----+
+END+
+-----+

```

RPM Calculation Instruction List Program

File 2, Rung 0

Ensures that the measurement value is initialized each RRUN mode entry.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|----------------|--------|
| 20 | -] [- | LD | 1'st pass S1/15 | 0 | |
| 106 | | MOV | SRC C0.ACC last timeout value storage register DEST N0 | 0000H 0000H | |
| 7 | | RES | C0 | | |
| 85 | | CLR | Counts last rate measurement period DEST N1 | 0000H | |
| 85 | | CLR | Frequency in Hertz DEST N4 | 0000H | |
| 85 | | CLR | RPM based on counts per turn register N7:2 DEST N5 | 0000H | |

File 2, Rung 1

Sets the rate measurement period. In this case, we are calculating a new rate value once every 100ms. Value N7:1 is updated once every 100ms with the number of counts that have occurred in the last 100ms period. Note that the preset value must divide evenly into 100 in order to accurately determine frequency and RPM (determined later in this program).

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|---------------------|--------|
| 21 | -] [- | LDI | Rate Period Expiration Bit T0/DN | 0 | |
| 0 | | TON | Counts last rate measurement period Rate Period Expiration Bit TIMR T0 BASE PRE ACC | 0.01 10 0000H | |

File 2, Rung 2

Calculates and stores the number of counts that have occurred since the last time that it was executed as true in N7:1 (last time=last rate measurement timer (T4:0) expiration). The LES instruction allows for 10 counts of backlash to occur (you can adjust as needed). The add instruction is configured for a 1000 count encoder using N7:2. (Change this register to match the number of counts generated each Z reset.)

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|-------------|--|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -] [- | LD | Rate | Period Expiration Bit T0/DN | 0 | |
| 10 | | MPS | | | | |
| 81 | | SUB | SRCA C0.ACC | last timeout value | 0000H | |
| | | | SRCB N0 | storage register | 0000H | |
| 11 | | MRD | | Counts last rate measurement period | | |
| | | | DEST N1 | | 0000H | |
| 22 | -] [- | AND | | If Negative Math Flag | | |
| | | | S0/3 | | 0 | |
| 57 | -LES- | AND-LES | | Counts last rate measurement period | | |
| | | | SRCA N1 | | 0000H | |
| | | | SRCB | | -10 | |
| 80 | | ADD | SRCA N2 | | 1000 | |
| | | | | Counts last rate measurement period | | |
| | | | SRCB N1 | | 0000H | |
| | | | | Counts last rate measurement period | | |
| | | | DEST N1 | | 0000H | |
| 11 | | MRD | | | | |
| 106 | | MOV | SRC C0.ACC | last timeout value | 0000H | |
| | | | DEST N0 | storage register | 0000H | |
| 11 | | MRD | | | | |
| 83 | | DIV | SRCA | | 100 | |
| | | | SRCB T0.PRE | | 10 | |
| | | | | Determine 1 second count. ie.# of rate periods | | |
| | | | DEST C1.PRE | | 10 | |
| 11 | | MRD | | | | |
| 5 | | CTU | | Frequency determination counter | | |
| | | | CNTR C1 | | | |
| | | | PRE | | 10 | |
| | | | ACC | | 0000H | |
| 11 | | MRD | | | | |
| 80 | | ADD | | Counts last rate measurement period | | |
| | | | SRCA N1 | | 0000H | |
| | | | | Frequency calculation register | | |
| | | | SRCB N3 | | 0000H | |
| | | | | Frequency calculation register | | |
| | | | DEST N3 | | 0000H | |
| 12 | | MPP | | | | |
| 22 | -] [- | AND | | 1 second has now elapsed | | |
| | | | C1/DN | | 0 | |

Appendix E
Application Example Programs

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|---------------|--------|
| 106 | | MOV | Frequency calculation register SRC N3 | 0000H | |
| | | | Frequency in Hertz DEST N4 | 0000H | |
| 85 | | CLR | Frequency calculation register DEST N3 | 0000H | |
| 7 | | RES | Frequency determination counter C1 | | |
| 82 | | MUL | Frequency in Hertz SRCA N4 SRCB | 0000H 60 | |
| | | | Temporary reg. (math reg is real destination) DEST N6 | 0000H | |
| 84 | | DDV | SRC N2 RPM based on counts per turn register N7:2 DEST N5 | 1000 0000H | |
| 42 | -(U)- | RST | Math overflow error bit S5/0 | 0 | |

File 2, Rung 3

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|-------------------------------|-------------------------------|--------|
| 170 | | HSC | TYPE CNTR C0 PRE ACC | Up (Res,Hld) 1000 0000H | |

On/Off Circuit Application Example

The following application example illustrates how to use an input to toggle an output either on or off. For a detailed explanation of:

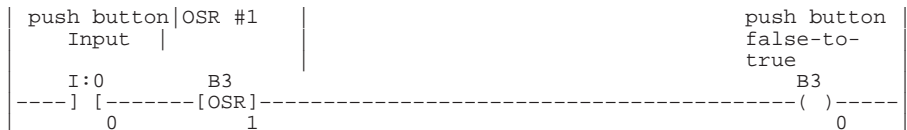
- LD, LDI, OUT, SET, RST, and OSR instructions, see chapter 8.
- JMP and LBL instructions, see chapter 12.

If the output is off when the input is energized, the output is turned on. If the output is on when the input is energized, the output is turned off.

On/Off Circuit Ladder Program

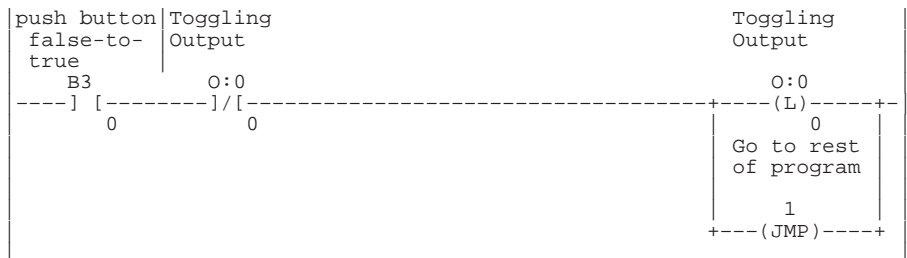
Rung 2:0

Does a one-shot from the input push button to an internal bit - the internal bit is true for only one scan. This prevents toggling of the physical output in case the push button is held "ON" for more than one scan (always the case).



Rung 2:1

If the push button input has gone from false-to-true and the output is presently OFF, turn the output ON and jump over the following rung to the rest of the programs. If the JMP instruction was missing, the following rung would be true and would turn the output back OFF.



Rung 2:2

If the push button input has gone from false-to-true and the output is presently ON, turns the output OFF.

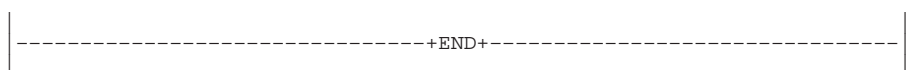


Rung 2:3

Contains the label corresponding to the jump instruction in rung 1. The remainder of your actual program is placed below this rung.



Rung 2:4



On/Off Circuit Instruction List Program

File 2, Rung 0

Does a one-shot from the input push button to an internal bit - the internal bit is true for only one scan. This prevent toggling of the physical output in case the push button is held "ON" for more than one scan (always the case).

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|--------------------------|--------------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -] [- | LD | Pushbutton | Input I/0 | 0 | |
| 29 | -OSR- | AND-OSR | OSR #1 | B/1 | 0 | |
| 40 | -()- | OUT | Pushbutton false-to-true | B/0 | 0 | |

File 2, Rung 1

If the push button input has gone from false-to-true and the output is presently OFF, turn the output ON and jump over the following rung to the rest of the programs. If the JMP instruction was missing, the following rung would be true and would turn the output back OFF.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|--------------------------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -] [- | LD | Pushbutton false-to-true | B/0 | 0 | |
| 23 | -] / [- | ANI | Toggling Output | O/0 | 0 | |
| 41 | -(L)- | SET | Toggling Output | O/0 | 0 | |
| 130 | | JMP | Go to rest of program | LBL# | 1 | |

File 2, Rung 2

If the push button input has gone from false-to-true and the output is presently ON, turns the output OFF.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|--------------------------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -] [- | LD | Pushbutton false-to-true | B/0 | 0 | |
| 22 | -] [- | AND | Toggling Output | O/0 | 0 | |
| 42 | -(U)- | RST | Toggling Output | O/0 | 0 | |

File 2, Rung 3

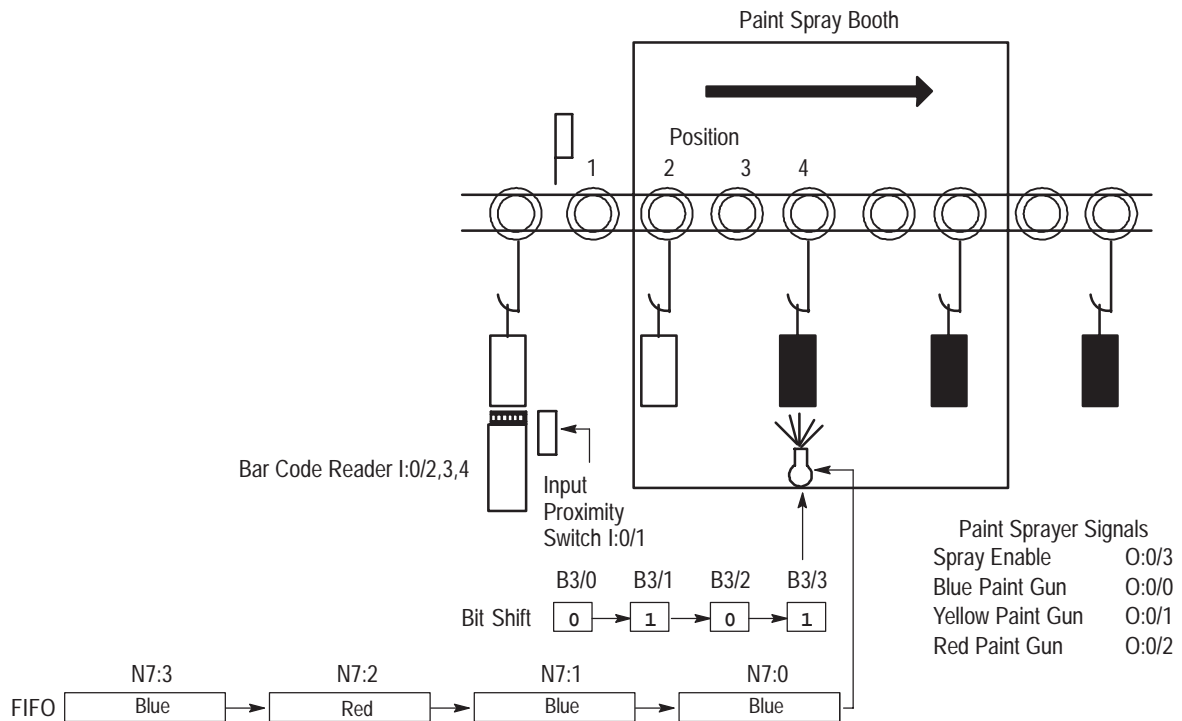
Contains the label corresponding to the jump instruction in rung 1. The remainder of your actual program is placed below this rung.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|-------------------------------|-------|--------|
| 131 | -LBL- | LD-LBL | Go to rest of program LBL# | 1 | |
| 40 | - () - | OUT | Dummy Bit B/2 | 0 | |

Spray Booth Application Example

The following application example illustrates the use of bit shift and FIFO instructions in an automated paint spraying operation. For a detailed explanation of:

- LD and OUT instructions, see chapter 8.
- EQU and LIM instructions, see chapter 9.
- FFU and FFL instructions, see chapter 11.
- BSL instruction, see chapter 12.



Spray Booth Operation Overview

An overhead conveyor with part carriers (hooks) carries parts from a previous operation to the spray booth. Before the part enters the spray booth, two items are checked on the conveyor. The first check is for part presence and the second check is for the needed color. This information is stored and accessed later when the part carrier is in the paint spraying area. A proximity switch is used to check for the presence of a part on the carrier and a barcode reader is used to determine color choice. When the part carrier reaches the spraying area, the previously stored information is accessed. If there is a part on the carrier, it is painted according to its bar code and if the carrier is free, paint is not dispensed.

The bit shift and FIFO instructions store the part presence and color information before each carrier enters the spray booth. Both of these instructions place data into their data structures every time a part carrier actuates the shift limit switch.

If the proximity switch senses a part on the carrier, a 1 is shifted into the shift register. If the carrier is free as it passes the shift limit switch, a 0 is shifted into the shift register. The shift register tracks the part carriers approaching the spraying area.

The FIFO does the same type of shifting, except rather than shifting one bit at a time, the FIFO shifts an entire word at a time. Just before the part carrier actuates the SHIFT limit switch, the barcode reader reads the barcode on the part to determine what color the part should be painted. The barcode reader has three outputs that it sets according to what color the part should be.

These outputs are:

- wired to the controller as inputs I/2, I/3, and I/4
- combined together to form an integer which is decoded later in the program

This integer is then shifted into the FIFO when the carrier actuates the SHIFT limit switch.

Once the presence and color data is loaded into the shift register and FIFO, they are shifted to new memory locations each time another part carrier actuates the SHIFT limit switch. After three additional shifts, the first part carrier is in front of the spray guns, ready for its part to be painted. At this point the part presence data has been shifted into B/3 and the color data has been shifted into N0. The program now checks B/3 – if there is a “1” in this location, then there is a part hanging on the part carrier and the SPRAY ENABLE output is energized. The program also checks N0 to determine which color to paint the part. As the program is checking the shift register for the presence of a part at the spray guns, it is also decoding the color information at N0 and energizing the appropriate spray guns. Since we are only using three colors, the only valid color codes are 1, 2, and 3. If any other number is in N0 when a part is ready to be painted, the color defaults to BLUE.

Since our program accesses the data while it is still in the two data structures, after the part has been painted, the presence and color information for that part is shifted out of the data structures and lost.

Spray Booth Ladder Program

Rung 2:0

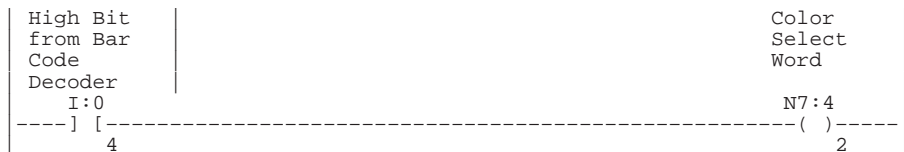
These three rungs read the color information coming from the barcode decoder outputs and load this into integer N7:4. This color is loaded into the FIFO when the part carrier actuates the SHIFT LIMIT SWITCH.



Rung 2:1

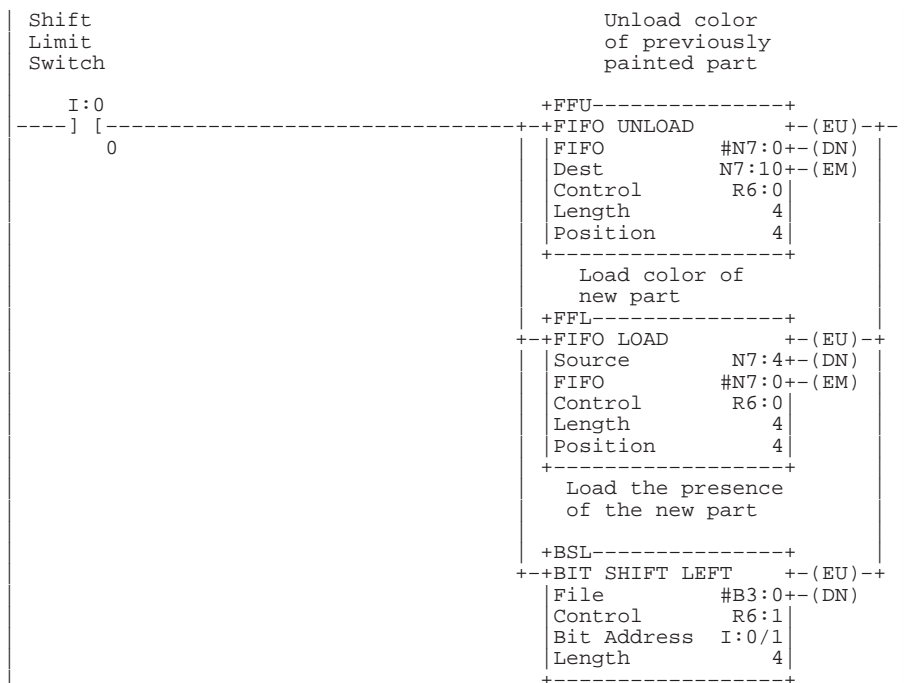


Rung 2:2



Rung 2:3

When the part carrier actuates the SHIFT LIMIT SWITCH, three things happen in this rung: (1) the color of the previously painted part is unloaded from the FIFO to make room for the color of the new part, (2) the color of the new part is loaded into the FIFO, (3) the presence or absence of a part on the part carrier is shifted into the Shift Register.



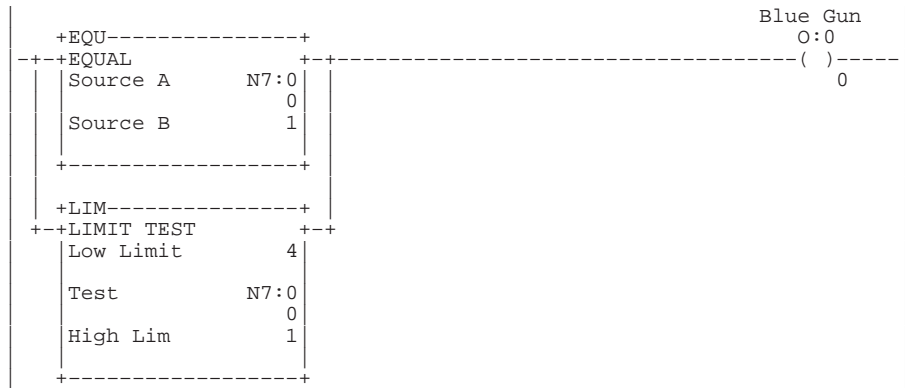
Rung 2:4

If there is a part on the part carrier now entering the spraying area, energize the paint sprayer. If there is not a part on the part carrier, do not energize the sprayer.



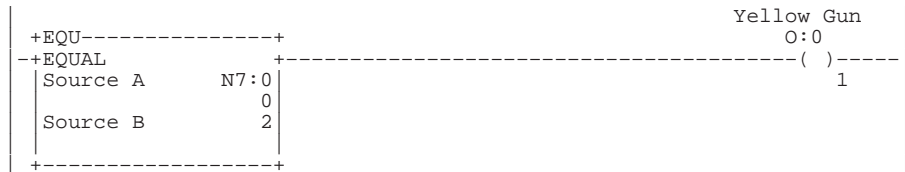
Rung 2:5

Decodes color select word. If N7:0=1, then energize the blue paint gun. Or if N7:0= an invalid color selection, default the color of the part to blue and energize the blue paint gun.



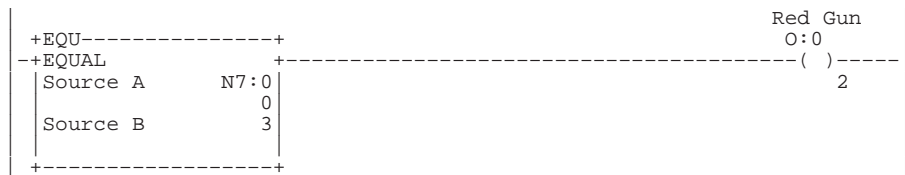
Rung 2:6

Decodes color select word. If N7:0=2, then energize the yellow paint gun.

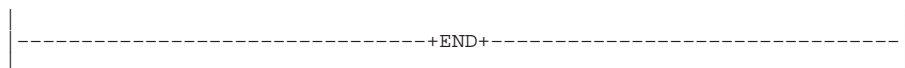


Rung 2:7

Decodes color select word. If N7:0=3, then energize the red paint gun.



Rung 2:8



Spray Booth Instruction List Program

File 2, Rung 0

These three rungs read the color information coming from the barcode decoder outputs and load this into integer N7:4. This color is loaded into the FIFO when the part carrier actuates the SHIFT LIMIT SWITCH.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--------------------------------------|-------|--------|
| 20 | -] [- | LD | Low Bit from Bar Code Decoder I/2 | 0 | |
| 40 | - () - | OUT | Color Select Word N4/0 | 0 | |

File 2, Rung 1

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|---|-------|--------|
| 20 | -] [- | LD | Middle Bit from Bar Code Decoder I/3 | 0 | |
| 40 | - () - | OUT | Color Select Word N4/1 | 0 | |

File 2, Rung 2

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--------------------------------------|-------|--------|
| 20 | -] [- | LD | Low Bit from Bar Code Decoder I/4 | 0 | |
| 40 | - () - | OUT | Color Select Word N4/2 | 0 | |

File 2, Rung 3

When the part carrier actuates the SHIFT LIMIT SWITCH, three things happen in this rung: (1) the color of the previously painted part is unloaded from the FIFO to make room for the color of the new part, (2) the color of the new part is loaded into the FIFO, (3) the presence or absence of a part on the part carrier is shifted into the Shift Register.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER NAME ADDRESS | VALUE | FORCES |
|-------------|-------------------|----------|--|-------|--------|
| 20 | -] [- | LD | Shift Limit Switch I/0 | 0 | |
| 114 | | FFU | Unload color of previously painted part FIFO #N0 DEST N10 CTRL R0 LEN 4 POS 4 | | |
| 113 | | FFL | Load color of new part SRC N4 FIFO #N0 CTRL R0 LEN 4 POS 4 | | |

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|-----------------------------------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 150 | | BSL | Load the presence of the new part | | | |
| | | | FILE #B0 | | | |
| | | | CTRL R1 | | | |
| | | | BIT I/1 | | | |
| | | | LEN | | 4 | |

File 2, Rung 4

If there is a part on the part carrier now entering the spraying area, energize the paint sprayer. If there is not a part on the part carrier, do not energize the sprayer.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|----------------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -] [- | LD | BSL Position 4 | | | |
| | | | B/3 | | 0 | |
| 40 | - () - | OUT | Spray Enable | | | |
| | | | O/3 | | 0 | |

File 2, Rung 5

Decodes color select word. If N7:0=1, then energize the blue paint gun. Or if N7:0= an invalid color selection, default the color of the part to blue and energize the blue paint gun.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|-----------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 50 | - EQU - | LD-EQU | SRCA N0 | | 0000H | |
| | | | SRCB | | 1 | |
| 73 | _ LIM _ | OR-LIM | LOW | | 4 | |
| | | | TEST N0 | | 0000H | |
| | | | HIGH | | 1 | |
| 40 | - () - | OUT | Blue Gun | | | |
| | | | O/0 | | 0 | |

File 2, Rung 6

Decodes color select word. If N7:0=2, then energize the yellow paint gun.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|------------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 50 | - EQU - | LD-EQU | SRCA N0 | | 0000H | |
| | | | SRCB | | 2 | |
| 40 | - () - | OUT | Yellow Gun | | | |
| | | | O/1 | | 0 | |

File 2, Rung 7

Decodes color select word. If N7:0=3, then energize the red paint gun.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|-----------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 50 | - EQU - | LD-EQU | SRCA N0 | | 0000H | |
| | | | SRCB | | 3 | |
| 40 | - () - | OUT | Red Gun | | | |
| | | | O/2 | | 0 | |

Adjustable Timer Application Example

The following application example illustrates the use of timers to adjust the drill dwell time at the end of the machines downstroke. For a detailed explanation of:

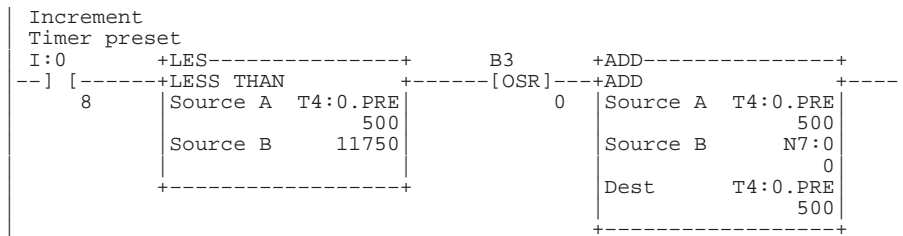
- LD, TON, and OSR instructions, see chapter 8.
- LES and GRT instructions, see chapter 9.
- ADD and SUB instructions, see chapter 10.

Valid dwell times are 5.0 seconds to 120.0 seconds. Adjustments are made in 2.5 second intervals.

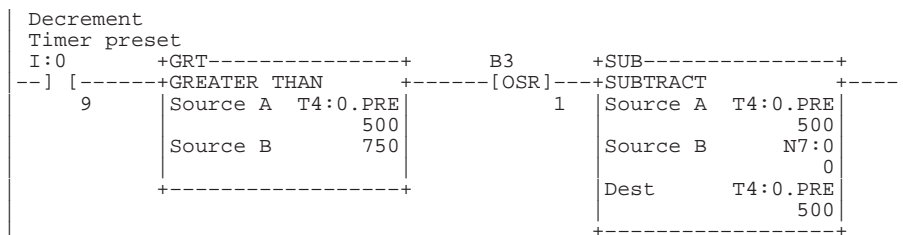
Each time I/8 or I/9 is depressed, the timer preset or delay is adjusted up or down accordingly. By altering the value of N0 the amount of change can be increased or decreased. The constants in the LES and GRT instructions, and in the source and destination of the ADD and SUB instructions, could be changed easily to integers for even greater flexibility.

Adjustable Timer Ladder Program

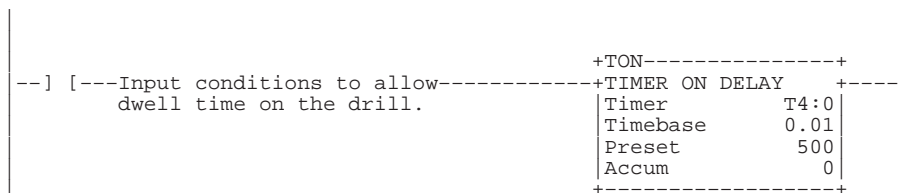
Rung 2:0
Adds 2.5 seconds to Timer delay each time the increment push button is depressed. Do not exceed 120.0 seconds delay. Note that N7:0=250.



Rung 2:1
Subtracts 2.5 seconds from Timer delay each time the decrement push button is depressed. Do not go below 5.0 seconds delay.



Rung 2:2



Adjustable Timer Instruction List Program

File 2, Rung 0

Adds 2.5 to Timer delay each time the increment push button is depressed. Do not exceed 120.0 seconds delay. Note that N7:0=250.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|------------------------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -] [- | LD | Increment Timer preset | | | |
| | | | | I/8 | 0 | |
| 57 | -LES- | AND-LES | SRCA | T0.PRE | 500 | |
| | | | SRCB | | 11750 | |
| 29 | -OSR- | AND-OSR | | B/0 | 0 | |
| 80 | | ADD | SRCA | T0.PRE | 500 | |
| | | | SRCB | N0 | 0000H | |
| | | | DEST | T0.PRE | 500 | |

File 2, Rung 1

Subtracts 2.5 seconds from Timer delay each time the decrement push button is depressed. Do not go below 5.0 seconds delay.

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|------------------------|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -] [- | LD | Decrement Timer preset | | | |
| | | | | I/9 | 0 | |
| 63 | -GRT- | AND-GRT | SRCA | T0.PRE | 500 | |
| | | | SRCB | | 750 | |
| 29 | -OSR- | AND-OSR | | B/1 | 0 | |
| 81 | | SUB | SRCA | T0.PRE | 500 | |
| | | | SRCB | N0 | 0000H | |
| | | | DEST | T0.PRE | 500 | |

File 2, Rung 2

| FUN CODE | GRAPHIC SYMBOL | MNEMONIC | PARAMETER | | VALUE | FORCES |
|-------------|-------------------|----------|--|---------|-------|--------|
| ----- | ----- | ----- | NAME | ADDRESS | ----- | ----- |
| 20 | -] [- | LD | Input conditions to allow dwell time on the drill. | | | |
| 0 | | TON | TIMR | T0 | | |
| | | | BASE | | 0.01 | |
| | | | PRE | | 500 | |
| | | | ACC | | 0000H | |

Optional Analog Input Software Calibration

This appendix helps you calibrate an analog input channel using software offsets to increase the expected accuracy of an analog input circuit. Examples of equations and a ladder diagram are provided for your reference. Software calibration reduces the error at a given temperature by scaling the values read at calibration time.

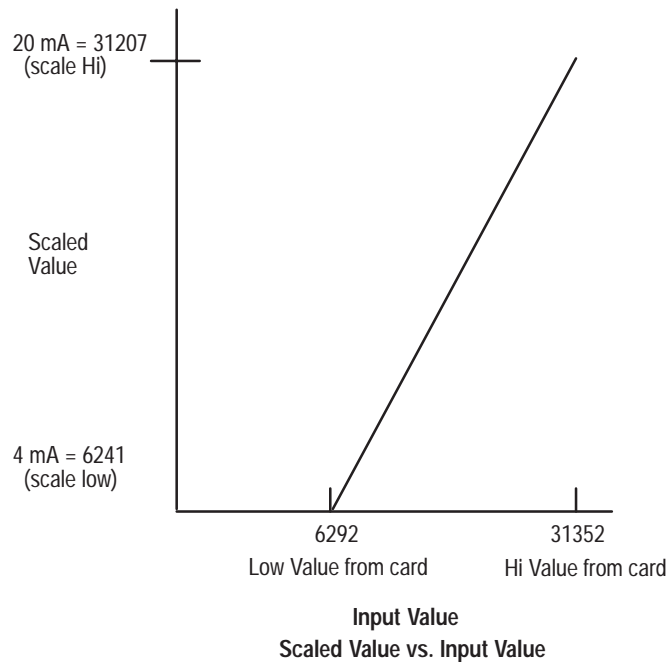
Calibrating an Analog Input Channel

The following procedure can be adapted to all analog inputs; current or voltage. For this example, the 1761-L20BWA-5A with a 4 mA to 20 mA input is used. The overall error for the MicroLogix 1000 is guaranteed to be not more than ± 0.525 at 25 °C.

The overall error of $\pm 0.525\%$ at 20 mA equates to ± 164 LSB of error, or a code range of 31043 to 31371. Any value in this range is returned by an analog input channel at 20 mA. The expected nominal value at 20 mA is 31207. After performing a software calibration, the overall error is reduced to 5 LSB (0.018%), or a code range of 31202 to 31212.

The graph shown below shows the linear relationship between the input value and the resulting scaled value. The values in this graph are from the example program.

Figure A.1



Calculating the Software Calibration

Use the following equation to perform the software calibration:

$$\text{Scaled Value} = (\text{input value} \times \text{slope}) + \text{offset}$$

$$\text{Slope} = (\text{scaled max.} - \text{scaled min.}) / (\text{input max.} - \text{input min.})$$

$$\text{Offset} = \text{Scaled min.} - (\text{input min.} \times \text{slope})$$

Calibration Procedure

1. Heat up / cool down your MicroLogix 1000 system to the temperature in which it will normally be operating.
2. Determine the scaled high and low values you wish to use in your application. In this example, scaled high value (which corresponds to 20 mA) is 31207 and scaled low value (which corresponds to 4 mA) is 6241.
3. Using an analog calibration source connected to the analog input channel or your system's input device placed at the 4 mA position, capture the low value by setting and then resetting the CAL_LO_ENABLE bit. Ensure that your low value lies within the conversion range of your analog input.
4. Using an analog calibration source connected to the analog input channel or your system's input device placed at the 20 mA position, capture the high value by setting and then resetting the CAL_HI_ENABLE bit. Ensure that your high value lies within the conversion range of your analog input.
5. Set and then reset the CALIBRATE bit. This causes the MicroLogix to calculate the slope and offset values used to perform the error correction to the analog input.

The analog channel is now calibrated to ± 5 LSB at the calibration temperature.

The recommended calibration period is once every 6 months. If an application has a wide range of operating temperatures, a software calibration should be performed every 3 to 4 months.

Example Ladder Diagram

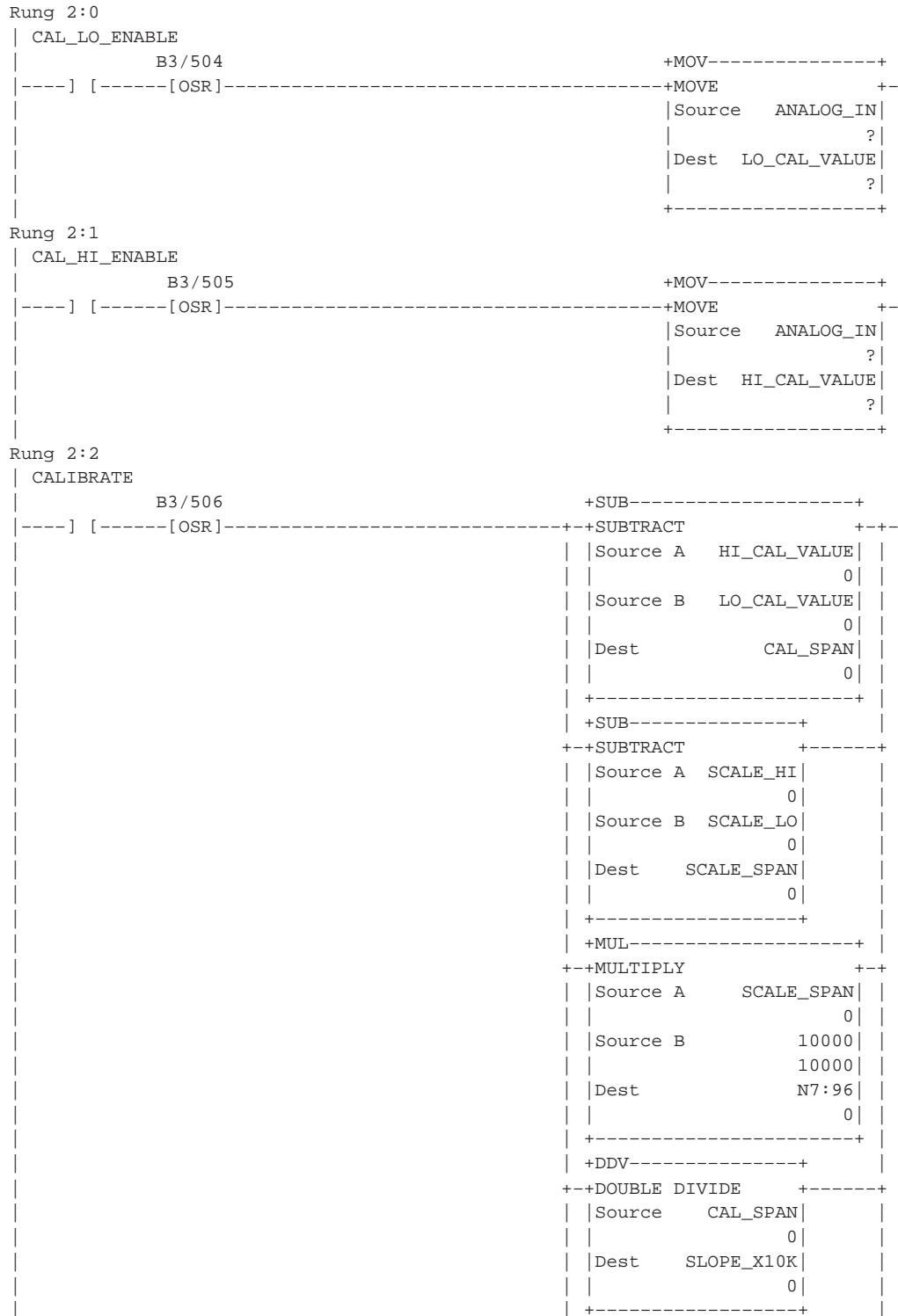
The following ladder diagram uses three internal bits to perform the calibration procedure. CAL_LO_ENABLE causes the ladder to capture the 4 mA calibration value and CAL_HI_ENABLE causes the ladder to capture the 20 mA calibration value. CALIBRATE causes the ladder diagram to scale the high and low values to the nominal values, which provides the slope and offset values used to calibrate the analog input channel.

Once the calibration procedure is complete, set the CONVERSION ENABLE bit to 1. The calibration numbers are then used to scale the raw analog data. The corrected analog input data is placed in memory location ANALOG_SCALED.

The following symbols are used in this example:

| | |
|-------------------|----------|
| CAL_LO_ENABLE | = B3/500 |
| CAL_HI_ENABLE | = B3/501 |
| CALIBRATE | = B3/502 |
| CONVERSION ENABLE | = B3/503 |
| ANALOG_IN | = I:0.4 |
| LO_CAL_VALUE | = N7:90 |
| HI_CAL_VALUE | = N7:91 |
| CAL_SPAN | = N7:92 |
| SCALE_HI | = N7:93 |
| SCALE_LOW | = N7:94 |
| SCALE_SPAN | = N7:95 |
| SLOPE_X10K | = N7:97 |
| OFFSET | = N7:100 |
| ANALOG_SCALED | = N7:101 |

Appendix F
Optional Analog Input Software Calibration



Ladder logic continued on the next page.

```

| +MUL-----+ |
+--MULTIPLY      +-+
| Source A   LO_CAL_VALUE |
|           0 |
| Source B   SLOPE_X10K |
|           0 |
| Dest      N7:98 |
|           0 |
+-----+
| +DDV-----+
+--DOUBLE DIVIDE  +-----+
| Source      10000 |
|           10000 |
| Dest      N7:99 |
|           0 |
+-----+
| +SUB-----+
+--SUBTRACT      +-----+
| Source A SCALE_LOW |
|           0 |
| Source B   N7:99 |
|           0 |
| Dest      OFFSET |
|           0 |
+-----+
|                                     Overflow |
|                                     Trap      |
|                                     S2:5/0   |
+-----+(U)-----+

Rung 2:3

CONVERSION_ENABLE      +SCL-----+
] [-----+SCALE      +-+
| Source A   ANALOG_IN |
|           ? |
| Rate[/10000] SLOPE_X10K |
|           ? |
| Offset      OFFSET |
|           ? |
| Dest      ANALOG_SCALED |
+-----+

+END+

```


Glossary

The following terms are used throughout this manual. Refer to the *Allen–Bradley Industrial Automation Glossary*, Publication Number AG–7.1, for a complete guide to Allen–Bradley technical terms.

address: A character string that uniquely identifies a memory location. For example, I1/0 is the memory address for the data located in the Input file location word1, bit 0.

application: 1) A machine or process monitored and controlled by a controller. 2) The use of computer– or processor–based routines for specific purposes.

backup data: Data saved to the controller with the program.

baud rate: The speed of communication between devices on a network. All devices must communicate at the same baud rate.

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

block diagrams: A schematic drawing.

Boolean operators: Logical operators such as AND, OR, NEG, NOT, and XOR that can be used singularly or in combination to form logic statements or circuits. Can have an output response be true or false.

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

comment: Text included with a program to explain what the program is doing. Comments do not affect the operation of the program in any way.

communication scan: A part of the controller’s operating cycle. Communication with other devices, such as programming software on a personal computer, takes place.

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

controller overhead: An internal portion of the operating cycle used for housekeeping and setup purposes.

control profile: The means by which a controller determines which outputs turn on under what conditions.

counter: 1) An electro-mechanical relay-type device that counts the occurrence of some event. May be pulses developed from operations such as switch closures, interruptions of light beams, or other discrete events.
2) In controllers a software counter eliminates the need for hardware counters. The software counter can be given a preset count value to count up or down whenever the counted event occurs.

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

data table: The part of the controller memory that contains I/O values and files where data is monitored, manipulated, and changed for control purposes.

DIN rail: Manufactured according to Deutsche Industrie Normenausschuss (DIN) standards, a metal railing designed to ease installation and mounting of your controller.

DOS: Disk Operating System. The operating system used to operate a personal computer.

DTE (Data Terminal Equipment): Equipment that is attached to a network to send or receive data, or both.

edit: To create or modify a ladder or instruction list program.

EMI: Electromagnetic interference.

encoder: 1) A rotary device that transmits position information. 2) A device that transmits a fixed number of pulses for each revolution.

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

FIFO (First-In-First-Out): The order that data is entered into and retrieved from a file.

file: A collection of information organized into one group.

floppy disk: A thin flexible disk, coated with magnetic oxide and used to store data.

full-duplex: A bidirectional mode of communication where data may be transmitted and received simultaneously (contrast with half-duplex).

half-duplex: A communication link in which data transmission is limited to one direction at a time.

HHP (Hand-Held Programmer): A device used to monitor and develop control logic programs for the micro controller.

high byte: Bits 8-15 of a word.

input device: A device, such as a push button or a switch, that supplies signals through input circuits to the controller.

inrush current: The temporary surge current produced when a device or circuit is initially energized.

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

Instruction List program: A program written in a list format using mnemonics. The program is used by a programmable controller to control devices.

instruction set: The set of general purpose instructions available with a given controller.

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

jump: Change in normal sequence of program execution, by executing an instruction that alters the program counter (sometimes called a branch). In ladder programs a JUMP (JMP) instruction causes execution to jump to a labeled rung.

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

LED (Light Emitting Diode): Used as status indicator for controller functions and inputs and outputs.

LIFO (Last-In-First-Out): The order that data is entered into and retrieved from a file.

low byte: Bits 0–7 of a word.

logic: A process of solving complex problems through the repeated use of simple functions that can be either true or false. General term for digital circuits and programmed instructions to perform required decision making and computational functions.

Master Control Relay (MCR): A mandatory hardwired relay that can be de-energized by any series-connected emergency stop switch. Whenever the MCR is de-energized, its contacts open to de-energize all application I/O devices.

mnemonic: A simple and easy to remember term that is used to represent a complex or lengthy set of information.

modes: Selected methods of operation. Example: RRUN, RCSN, RSSN, or RPRG.

negative logic: The use of binary logic in such a way that “0” represents the voltage level normally associated with logic 1 (for example, 0 = +5V, 1 = 0V). Positive is more conventional (for example, 1 = +5V, 0 = 0V).

network: A series of stations (nodes) connected by some type of communication medium. A network may be made up of a single link or multiple links.

nominal input current: The current at nominal input voltage.

normally closed: Contacts on a relay or switch that are closed when the relay is de-energized or the switch is de-activated; they are open when the relay is energized or the switch is activated. In Instruction List programming, a symbol that allows logic continuity (flow) if the referenced input is logic “0” when evaluated.

normally open: Contacts on a relay or switch that are open when the relay is de-energized or the switch is de-activated. (They are closed when the relay is energized or the switch is activated.) In Instruction List programming, a symbol that allows logic continuity (flow) if the referenced input is logic “1” when evaluated.

one-shot: A programming technique that sets a bit for only one program scan.

online: Describes devices under direct communication. For example, when programming software is monitoring the program file in a controller.

operating voltage: For inputs, the voltage range needed for the input to be in the On state. For outputs, the allowable range of user-supplied voltage.

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

processor: A Central Processing Unit. (See CPU.)

program: The set of program and data files used by the controller to control output devices. Only one program may be stored in the controller at a time.

program data: Provides data locations for output, input, status, bit, timer, counter, control, and integer files.

program file: The area within a program that contains controller information, the main program, interrupt subroutines, and any subroutine programs.

program listing: A report containing a range of program files or a range of rungs.

program mode: When the controller is not executing the program and all outputs are de-energized.

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

protocol: The packaging of information that is transmitted across a network.

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

relay: An electrically operated device that mechanically switches electrical circuits.

relay logic: A representation of the program or other logic in a form normally used for relays.

restore: To download (transfer) a program from a personal computer to a controller.

reserved bit: A status file location that the user should not read or write to.

retentive data: Information associated with data files (timers, counters, inputs, and outputs) in a program that is preserved through power cycles. Program files 2–15 are not effected by retentive data.

RRUN mode: remote mode during which the controller scans or executes the logic program, monitors input devices, energizes output devices, and acts on enabled I/O forces.

RS–232: An EIA standard that specifies electrical, mechanical, and functional characteristics for serial binary communication circuits. A single-ended serial communication interface.

run mode: When the program in the controller is being executed, inputs are read, the program is scanned, and outputs are energized and de-energized.

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

scan time: The time required for the controller to execute the instructions in the program. The scan time may vary depending on the instructions and each instruction's status during the scan.

sinking: A term used to describe current flow between an I/O device and controller I/O circuit — typically, a sinking device or circuit provides a path to ground, low, or negative side of power supply.

software: Executable programming package used to develop ladder diagrams.

sourcing: A term used to describe current flow between an I/O device and controller I/O circuit — typically, a sourcing device or circuit provides a path to the source, high, or positive side of power supply.

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

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

throughput: The time between when an input turns on and the corresponding output turns on.

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

upload: Data is transferred to a programming or storage device from another device.

user interrupt poll: While executing the user program, the controller firmware checks for user interrupts that need servicing.

watchdog timer: A timer that monitors a cyclical process and is cleared at the conclusion of each cycle. If the watchdog runs past its programmed time period, it will cause a fault.

workspace: The main storage available for programs and data and allocated for working storage.

write: To copy data to a storage device. For example, the controller WRITES the information from the output data file to the output devices.

Numbers

- 1761-HHP-B30, features, 4-2
- 1761-L10BWA
 - features, 1-2
 - grounding, 2-1
 - input voltage range, 2-8
 - mounting, 1-12
 - output voltage range, 2-8
 - preventing excessive heat, 1-12
 - spacing, 1-12
 - troubleshooting, 20-1
 - type, 1-2
 - wiring, 2-3
 - wiring diagram, 2-8
- 1761-L10BWB
 - features, 1-2
 - grounding, 2-1
 - input voltage range, 2-11
 - mounting, 1-12
 - output voltage range, 2-11
 - preventing excessive heat, 1-12
 - spacing, 1-12
 - troubleshooting, 20-1
 - type, 1-2
 - wiring, 2-3
 - wiring diagram, 2-11
- 1761-L16AWA
 - features, 1-2
 - grounding, 2-1
 - input voltage range, 2-6
 - mounting, 1-12
 - output voltage range, 2-6
 - preventing excessive heat, 1-12
 - spacing, 1-12
 - troubleshooting, 20-1
 - type, 1-2
 - wiring, 2-3
 - wiring diagram, 2-6
- 1761-L16BBB
 - features, 1-2
 - grounding, 2-1
 - input voltage range, 2-15
 - mounting, 1-12
 - output voltage range, 2-15
 - preventing excessive heat, 1-12
 - spacing, 1-12
 - troubleshooting, 20-1
 - type, 1-2
 - wiring, 2-3
 - wiring diagram, 2-15
- 1761-L16BWA
 - features, 1-2
 - grounding, 2-1
 - input voltage range, 2-9
 - mounting, 1-12
 - output voltage range, 2-9
 - preventing excessive heat, 1-12
 - spacing, 1-12
 - troubleshooting, 20-1
 - type, 1-2
 - wiring, 2-3
 - wiring diagram, 2-9
- 1761-L16BWB
 - features, 1-2
 - grounding, 2-1
 - input voltage range, 2-12
 - mounting, 1-12
 - output voltage range, 2-12
 - preventing excessive heat, 1-12
 - spacing, 1-12
 - troubleshooting, 20-1
 - type, 1-2
 - wiring, 2-3
 - wiring diagram, 2-12
- 1761-L20AWA-5A
 - features, 1-2
 - grounding, 2-1
 - input voltage range, 2-17
 - mounting, 1-12
 - output voltage range, 2-17
 - preventing excessive heat, 1-12
 - spacing, 1-12
 - troubleshooting, 20-1
 - type, 1-2
 - wiring, 2-3
 - wiring diagram, 2-17
- 1761-L20BWA-5A
 - features, 1-2
 - grounding, 2-1
 - input voltage range, 2-18
 - mounting, 1-12
 - output voltage range, 2-18
 - preventing excessive heat, 1-12
 - spacing, 1-12
 - troubleshooting, 20-1
 - type, 1-2
 - wiring, 2-3
 - wiring diagram, 2-18
- 1761-L20BWB-5A
 - features, 1-2
 - grounding, 2-1
 - input voltage range, 2-19
 - mounting, 1-12
 - output voltage range, 2-19
 - preventing excessive heat, 1-12

- spacing, 1–12
 - troubleshooting, 20–1
 - type, 1–2
 - wiring, 2–3
 - wiring diagram, 2–19
 - 1761-L32AAA
 - features, 1–2
 - grounding, 2–1
 - input voltage range, 2–14
 - mounting, 1–12
 - output voltage range, 2–14
 - preventing excessive heat, 1–12
 - spacing, 1–12
 - troubleshooting, 20–1
 - type, 1–2
 - wiring, 2–3
 - wiring diagram, 2–14
 - 1761-L32AWA
 - features, 1–2
 - grounding, 2–1
 - input voltage range, 2–7
 - mounting, 1–12
 - output voltage range, 2–7
 - preventing excessive heat, 1–12
 - spacing, 1–12
 - troubleshooting, 20–1
 - type, 1–2
 - wiring, 2–3
 - wiring diagram, 2–7
 - 1761-L32BBB
 - features, 1–2
 - grounding, 2–1
 - input voltage range, 2–16
 - mounting, 1–12
 - output voltage range, 2–16
 - preventing excessive heat, 1–12
 - spacing, 1–12
 - troubleshooting, 20–1
 - type, 1–2
 - wiring, 2–3
 - wiring diagram, 2–16
 - 1761-L32BWA
 - features, 1–2
 - grounding, 2–1
 - input voltage range, 2–10
 - mounting, 1–12
 - output voltage range, 2–10
 - preventing excessive heat, 1–12
 - spacing, 1–12
 - troubleshooting, 20–1
 - type, 1–2
 - wiring, 2–3
 - wiring diagram, 2–10
 - 1761-L32BWB
 - features, 1–2
 - grounding, 2–1
 - input voltage range, 2–13
 - mounting, 1–12
 - output voltage range, 2–13
 - preventing excessive heat, 1–12
 - spacing, 1–12
 - troubleshooting, 20–1
 - type, 1–2
 - wiring, 2–3
 - wiring diagram, 2–13
 - 32-bit addition and subtraction, 10–6
 - example, 10–6
 - instruction list program, 10–7
 - ladder representation, 10–7
 - math overflow selection bit S2/14, 10–6
- ## A
- accepting your program edits, 18–21
 - accessing additional characters, HHP keypad, 4–5
 - accessing the program
 - normal operation, 6–6
 - power up, 6–7
 - accessories and replacement parts, A–11
 - Add (ADD), 10–4
 - entering the instruction, 10–4
 - execution times, 10–4
 - function code, 10–4
 - instruction parameters, C–3
 - ladder representation, 10–4
 - updates to arithmetic status bits, 10–4
 - valid addressing modes, C–3
 - valid file types, C–3
 - ADD, Add, 10–4
 - addresses, searching for, 17–8
 - addressing
 - compared to programming software's addressing, 6–8
 - data files, 6–7
 - indexed, 6–9
 - logical addresses, specifying, 6–8
 - using mnemonics, 6–8
 - addressing modes, C–2
 - direct addressing, C–2
 - immediate addressing, C–2
 - indexed addressing, C–2

- AIC+
 - applying power to, 3–11
 - attaching to the network, 3–12
 - connecting, 3–6
 - modem, 3–7
 - network, 3–7
 - point-to-point, 3–7
 - installing, 3–12
 - recommended user supplied components, 3–10
 - selecting cable, 3–8
- Allen-Bradley, contacting for assistance, P–6, 20–16
- analog controller, types, 20 discrete I/O and 5 analog I/O, 1–2
- analog controllers
 - converting analog data, 7–4
 - grounding your analog cable, 2–20
 - I/O configuration, 7–2
 - I/O image, 7–1
 - input channel filtering, 7–3
 - input software calibration, F–1
 - minimizing electrical noise, 2–20
 - voltage and current input and output ranges, 2–22
 - wiring your analog channels, 2–21
- ANB, And Block, 8–12
- And (AND)
 - bit input instruction, 8–3
 - entering the instruction, 8–4
 - execution times, 8–3
 - instruction parameters, C–3
 - ladder representation, 8–3
 - using, 8–4
 - valid addressing modes, C–3
 - valid file types, C–3
 - word output instruction, 11–18
 - entering the instruction, 11–18
 - execution times, 11–18
 - function code, 11–18
 - instruction parameters, C–3
 - ladder representation, 11–18
 - updates to arithmetic status bits, 11–18
 - valid addressing modes, C–3
 - valid file types, C–3
- And Block (ANB), 8–12
 - entering the instruction, 8–12
 - execution times, 8–12
 - function code, 8–12, 8–13
 - ladder representation, 8–12
 - using, 8–12
- And Inverted (ANI), 8–4
 - entering the instruction, 8–5
 - execution times, 8–4
 - instruction parameters, C–3
 - ladder representation, 8–4
 - using, 8–5
 - valid addressing modes, C–3
 - valid file types, C–3
- AND, And
 - bit input instruction, 8–3
 - word output instruction, 11–18
- ANI, And Inverted, 8–4
- append mode, 17–3
 - inserting a rung, 17–4
 - inserting an instruction, 17–3
- application example programs
 - adjustable timer, E–55
 - instruction list program, E–56
 - ladder program, E–55
 - operation overview, E–55
 - bottle line, E–29
 - instruction list program, E–32
 - ladder program, E–29
 - operation overview, E–29
 - conveyor line, E–34
 - instruction list program, E–37
 - ladder program, E–35
 - operation overview, E–34
 - event driven sequencer, E–27
 - instruction list program, E–28
 - ladder program, E–27
 - operation overview, E–27
 - on/off circuit, E–47
 - instruction list program, E–48
 - ladder program, E–47
 - operation overview, E–47
 - paper drilling machine, E–2
 - instruction list program, E–14
 - ladder program, E–4
 - operation overview, E–2
 - RPM calculation, E–40
 - instruction list program, E–44
 - ladder program, E–41
 - operation overview, E–40
 - spray booth, E–49
 - instruction list program, E–53
 - ladder program, E–51
 - operation overview, E–50
 - time driven sequencer, E–25
 - instruction list program, E–26
 - ladder program, E–25
 - operation overview, E–25

- using the MSG instruction, 15–12
- application specific instructions
 - about, 13–1
 - bit shift instructions, overview, 13–2
 - Bit Shift Left (BSL), 13–3
 - Bit Shift Right (BSR), 13–4
 - in the paper drilling machine application
 - example, 13–20
 - Selectable Timed Interrupt (STI) function,
 - overview, 13–15
 - sequencer instructions, overview, 13–6
- applying logic to your schematics, 6–11

B

- basic instructions
 - about, 8–2
 - bit instructions, overview, 8–3
 - branch instructions, overview, 8–9
 - counter instructions, overview, 8–21
 - in the paper drilling machine application
 - example, 8–28
 - timer instructions, overview, 8–14
- baud rate
 - changing, 19–6, 19–7
 - limitations for autoswitching, 3–13
- bidirectional counter
 - operation, 14–9
 - overview, 14–5
- bidirectional counter with quadrature encoder
 - operation, 14–13
 - overview, 14–5
- bidirectional counter with reset and hold
 - operation, 14–9
 - overview, 14–5
- bidirectional counter with reset and hold with quadrature encoder
 - operation, 14–13
 - overview, 14–5
- bit file (B), 6–4
- bit instructions
 - And (AND), 8–3
 - And Block (ANB), 8–12
 - And Inverted (ANI), 8–4
 - Load (LD), 8–3
 - Load Inverted (LDI), 8–4
 - Load True (LDT), 8–6
 - Memory Pop (MPP), 8–10
 - Memory Push (MPS), 8–10
 - Memory Read (MRD), 8–10
 - One-Shot Rising (OSR), 8–7
 - Or (OR), 8–3

- Or Block (ORB), 8–12
- Or Inverted (ORI), 8–4
- Or True (ORT), 8–6
- Output (OUT), 8–8
 - overview, 8–3
- Reset (RST), 8–8
- Set (SET), 8–8
- bit shift instructions, overview, 13–2
 - effects on index register S24, 13–3
 - entering parameters, 13–2
 - entering the instructions, 13–2
- Bit Shift Left (BSL), 13–3
 - effects on index register, 13–3
 - entering parameters, 13–2
 - entering the instruction, 13–3
 - execution times, 13–3
 - function code, 13–3
 - instruction parameters, C–3
 - ladder representation, 13–3
 - operation, 13–4
 - valid file types, C–3
- Bit Shift Right (BSR), 13–4
 - effects on index register, 13–3
 - entering parameters, 13–2
 - entering the instruction, 13–4
 - execution times, 13–4
 - function code, 13–5
 - instruction parameters, C–3
 - ladder representation, 13–4
 - operation, 13–5
 - valid addressing modes, C–3
 - valid file types, C–3
- branch instructions, overview, 8–9
- branching
 - input, 6–13
 - nested, 6–14
 - output, 6–13
- BSL, Bit Shift Left, 13–3
- BSR, Bit Shift Right, 13–4

C

- cables
 - planning routes for DH-485 connections,
 - D–13
 - selection guide for the AIC+, 3–8
 - selection guide for the DeviceNet network,
 - 3–14
- Calculating the Software Calibration, F–2
- calibrating an analog input channel, F–1
- CE mark, 1–1

- changing
 - editing modes, 17–3
 - radix, 18–30
 - remote controller modes, 18–21, 18–23
 - the baud rate, 19–6, 19–7
 - the HHP's defaults, 4–17
 - the language, 4–17
 - the LCD display contrast, 4–18
 - the program defaults, 18–1
 - controller version, 18–18
 - extended I/O configuration bit, 18–8
 - fault override bit, 18–7
 - input filters, 18–12, 18–14, 18–15, 18–16
 - lock program function, 18–17
 - program name, 18–2
 - run always bit, 18–5
 - start-up protection bit, 18–6
 - STI enabled bit, 18–10
 - STI setpoint, 18–9
 - user and master passwords, 18–2
 - watchdog scan, 18–11
- channel configuration
 - DF1 full-duplex, D–2
 - DF1 half-duplex slave, D–5
- Clear (CLR), 10–11
 - entering the instruction, 10–11
 - execution times, 10–11
 - function code, 10–11
 - instruction parameters, C–3
 - ladder representative, 10–11
 - updates to arithmetic status bits, 10–11
 - valid addressing modes, C–3
 - valid file types, C–3
- clearing a program
 - from a memory module, 19–5
 - from the controller, 5–2, 19–6
- clearing faults, 20–12
- CLR, Clear, 10–11
- common procedures, 19–1
- communication
 - DeviceNet, 3–13
 - establishing with controller, 3–12, 3–13, 19–7
 - types of, 15–1
- communication protocols
 - DF1 full-duplex, D–2
 - DF1 half-duplex slave, D–4
 - DH-485, D–9
- comparison instructions, 9–1
 - about, 9–2
 - Equal (EQU), 9–3
 - Greater Than (GRT), 9–7
 - Greater Than or Equal (GEO), 9–8
 - Less Than (LES), 9–5
 - Less Than or Equal (LEQ), 9–6
 - Limit Test (LIM), 9–10
 - Masked Comparison for Equal (MEQ), 9–9
 - Not Equal (NEQ), 9–4
 - overview, 9–2
 - entering the instruction, 9–2
 - function codes, 9–3
 - using indexed word addresses, 9–3
- configuring your program, 18–1
 - controller version, 18–18
 - extended I/O configuration, 18–8
 - fault override bit, 18–7
 - input filter settings, 18–12, 18–14, 18–15, 18–16
 - lock program function, 18–17
 - program name, 18–2
 - run always bit, 18–5
 - start-up protection, 18–6
 - STI enabled bit, 18–10
 - STI setpoint, 18–9
 - user and master passwords, 18–2
 - watchdog scan, 18–11
- connecting blocks of logic
 - using ANB, 6–14, 8–12, 16–2
 - using ORB, 6–14, 8–12, 16–3
- connecting the system
 - AIC+, 3–6
 - DeviceNet network, 3–13
 - DH-485 network, 3–3
 - HHP, 3–1
- contact protection methods, 1–7
- contacting Allen-Bradley for assistance, P–6
- contactors (bulletin 100), surge suppressors for, 1–9
- contents of manual, P–2
- continuous scan (CSN) mode, 18–22
- control file (R), 6–4
- controller
 - accessories and replacement parts, A–11
 - changing modes, 18–21
 - changing the baud rate, 19–6, 19–7
 - determining faults, 20–1
 - dimensions, A–7
 - fault messages, 20–13
 - features, 1–2
 - grounding, 2–1
 - installation, 1–1
 - mounting, 1–12

- mounting template, A-7
 - operating cycle, 6-2
 - spacing, 1-12
 - specifications, A-1
 - status file, B-1
 - troubleshooting, 20-1
 - types, 1-2, A-1
 - 16 I/O, 1-2
 - 20 discrete I/O and 5 analog I/O, 1-2
 - 32 I/O, 1-2
 - wiring
 - for high-speed counter applications, 2-23
 - recommendations, 2-3
 - wire type, 2-3
 - controller faults, 20-1
 - controller modes
 - changing remote modes, 18-23
 - modes of operation, 18-22
 - tasks you can perform, 18-24
 - types of modes, 18-22
 - remote program mode, 18-22
 - remote run mode, 18-22
 - remote test mode, 18-22
 - controller operation, normal, 20-1
 - controller version, changing the version the HHP supports, 18-18
 - Convert from BCD (FRD), 11-3
 - entering the instruction, 11-3
 - example 1, 11-4
 - instruction list program, 11-5
 - ladder representation, 11-4
 - example 2, 11-5
 - instruction list program, 11-6
 - ladder representation, 11-6
 - execution times, 11-3
 - function code, 11-4
 - instruction parameters, C-5
 - ladder representation, 11-3
 - updates to arithmetic status bits, 11-3
 - valid addressing modes, C-5
 - valid file types, C-5
 - Convert to BCD (TOD), 11-2
 - changes to the math register, 11-3
 - entering the instruction, 11-2
 - execution times, 11-2
 - function code, 11-3
 - instruction parameters, C-10
 - ladder representation, 11-2
 - updates to arithmetic status bits, 11-2
 - valid addressing modes, C-10
 - valid file types, C-10
 - Converting Analog Input Data, 7-4
 - Converting Analog Output Data, 7-5
 - COP, Copy File, 11-10
 - Copy File (COP), 11-10
 - entering parameters, 11-10
 - entering the instruction, 11-11
 - execution times, 11-10
 - function code, 11-11
 - instruction parameters, C-3
 - ladder representation, 11-10
 - using, 11-10
 - valid addressing modes, C-3
 - valid file types, C-3
 - Count Down (CTD), 8-25
 - entering the instruction, 8-26
 - execution times, 8-25
 - function code, 8-26
 - instruction parameters, C-3
 - ladder representation, 8-25
 - using status bits, 8-26
 - valid addressing modes, C-3
 - valid file types, C-3
 - Count Up (CTU), 8-24
 - entering the instruction, 8-25
 - execution times, 8-24
 - function code, 8-25
 - instruction parameters, C-4
 - ladder representation, 8-24
 - using status bits, 8-24
 - valid addressing modes, C-4
 - valid file types, C-4
 - counter file (C), 6-4
 - counter instructions
 - Count Down (CTD), 8-25
 - Count Up (CTU), 8-24
 - in the paper drilling machine application
 - example, 9-12
 - overview, 8-21
 - addressing structure, 8-23
 - entering parameters, 8-22
 - entering the instructions, 8-22
 - how counters work, 8-24
 - Reset (RES), 8-27
 - CTD, Count Down, 8-25
 - CTU, Count Up, 8-24
- ## D
- data files
 - addressing, 6-7
 - organization, 6-4
 - types, 6-7
 - file indicator (#), 6-10

- data handling instructions
 - about, 11-2
 - Convert from BCD (FRD), 11-3
 - Convert to BCD (TOD), 11-2
 - Copy File (COP), 11-10
 - Decode 4 to 1 of 16 (DCD), 11-7
 - Encode 1 of 16 to 4 (ENC), 11-8
 - FIFO and LIFO instructions, overview, 11-23
 - Fill File (FLL), 11-10
 - in the paper drilling machine application
 - example, 11-31
 - move and logical instructions, overview, 11-13
- data monitor
 - description, 4-13
 - entering, 18-26
 - how to complete tasks, 4-14
 - screen definition, 4-13
- data table status file displays, 18-28
 - bit, 18-29
 - control, 18-29
 - counter, 18-29
 - input, 18-28
 - integer, 18-30
 - output, 18-28
 - status, 18-28
 - timer, 18-29
- DCD, Decode 4 to 1 of 16, 11-7
- DDV, Double Divide, 10-10
- Decode 4 to 1 of 16 (DCD), 11-7
 - entering parameters, 11-7
 - entering the instruction, 11-7
 - execution times, 11-7
 - function code, 11-8
 - instruction parameters, C-4
 - ladder representative, 11-7
 - updates to arithmetic status bits, 11-7
 - valid addressing modes, C-4
 - valid file types, C-4
- default settings, HHP's, 4-17
- deleting
 - a range of rungs, 17-7
 - a single rung, 17-6
 - an instruction, 17-6
 - multi-point addresses, 18-34
- determining controller faults, 20-2
- developing your logic program—a model, 6-17
- DeviceNet network
 - connecting, 3-13
 - selecting cable, 3-14
- DF1 full-duplex protocol
 - configuration parameters, D-2
 - description, D-2
 - example system configuration, D-3
 - using a modem, D-7
- DF1 half-duplex slave protocol, D-4
 - configuration parameters, D-5
 - using a modem, D-7
- DH-485 communication protocol,
 - configuration parameters, D-10
- DH-485 network
 - connecting, 3-3
 - description, D-9
 - devices that use the network, D-10
 - example system configuration, D-14
 - initialization, D-10
 - installation, 3-3
 - planning considerations, D-12
 - protocol, D-9
 - software considerations, D-13
 - token rotation, D-9
- diagnostic/troubleshooting keys, identifying, 4-4
- dimensions
 - controller, A-7
 - HHP, A-9
- DIN rail, 1-13
 - mounting dimensions, 1-13
- diode, 1N4004, 1-8
- direct addressing, C-2
- displaying values, 6-10
- DIV, Divide, 10-9
- Divide (DIV), 10-9
 - changes to the math register, 10-9
 - entering the instruction, 10-9
 - execution times, 10-9
 - function code, 10-9
 - instruction parameters, C-4
 - ladder representation, 10-9
 - updates to arithmetic status bits, 10-9
 - valid addressing modes, C-4
 - valid file types, C-4
- Double Divide (DDV), 10-10
 - changes to the math register, 10-10
 - entering the instruction, 10-10
 - execution times, 10-10
 - function code, 10-10
 - instruction parameters, C-4
 - ladder representative, 10-10
 - updates to arithmetic status bits, 10-10

valid addressing modes, C-4
valid file types, C-4

E

editing

accepting edits, 18-21
modes, 17-3
 append, 17-3
 overwrite, 17-4
searching for specific addresses, 17-8
your program, 17-1

editing considerations, 17-3

Electronics Industries Association (EIA), D-1

EMC Directive, 1-1

emergency-stop switches, 1-4

ENC, Encode 1 of 16 to 4, 11-8

Encode 1 of 16 to 4 (ENC), 11-8

entering parameters, 11-8
entering the instruction, 11-9
execution times, 11-8
function code, 11-9
instruction parameters, C-4
ladder representation, 11-8
updates to arithmetic status bits, 11-9
valid addressing modes, C-4
valid file types, C-4

End of File screen, 17-2

entering

and running a program, 5-4
 changing to run mode, 5-7
 entering the new program, 5-4
data monitor, 18-26
numeric constants, 6-10
program monitor, 17-1, 18-25
the # character, 6-9

EQU, Equal, 9-3

Equal (EQU), 9-3

AND EQU

entering the instruction, 9-3
execution times, 9-3
function code, 9-3
instruction parameters, C-4
ladder representation, 9-3
valid addressing modes, C-4
valid file types, C-4

LD EQU

entering the instruction, 9-3
execution times, 9-3
function code, 9-3
instruction parameters, C-4

ladder representation, 9-3
valid addressing modes, C-4
valid file types, C-4

OR EQU

entering the instruction, 9-3
execution times, 9-3
function code, 9-3
instruction parameters, C-4
ladder representation, 9-3
valid addressing modes, C-4
valid file types, C-4

error recovery model, 20-10

errors

download, B-10
going-to-run, B-9
hardware, 20-2
HHP
 communication, 20-4
 hardware, 20-3
 identifying, 20-3
 miscellaneous, 20-5
 program verification, 20-5
identifying, 20-11
MSG instruction, 15-11
powerup, B-9
run, B-10

establishing communication, 3-12, 19-7

European Union Directive compliance, 1-1

example programs

adjustable timer, E-55
bottle line, E-29
event driven sequencer, E-27
on/off circuit, E-47
paper drilling machine, E-2
pick and place machine, E-34
RPM calculation, E-40
spray booth, E-49
time driven sequencer, E-25
using the MSG instruction, 15-12

Exclusive Or (XOR), 11-20

entering the instruction, 11-20
execution times, 11-20
function code, 11-20
instruction parameters, C-10
ladder representation, 11-20
updates to arithmetic status bits, 11-20
valid addressing modes, C-10
valid file types, C-10

executing the ladder program, changing to
Run mode, 5-7

execution times

listing, B-16

worksheet, B-21
 extended I/O configuration bit, setting, 18-8

F

fault messages, 20-13
 fault override bit, setting, 18-7
 fault recovery procedure, 20-11, 20-12
 fault routine, 20-12
 FFL, FIFO Load, 11-25
 FFU, FIFO Unload, 11-25
 FIFO and LIFO instructions
 FIFO Load (FFL), 11-25
 FIFO Unload (FFU), 11-25
 LIFO Load (LFL), 11-28
 LIFO Unload (LFU), 11-28
 overview, 11-23
 effects on index register S24, 11-24
 entering parameters, 11-23
 entering the instructions, 11-24
 FIFO Load (FFL), 11-25
 entering the instruction, 11-25
 execution times, 11-25
 function code, 11-25
 instruction parameters, C-4
 ladder representation, 11-25, 11-28
 operation, 11-26
 valid addressing modes, C-4
 valid file types, C-4
 FIFO Unload (FFU), 11-25
 entering the instruction, 11-26
 execution times, 11-25
 function code, 11-26
 instruction parameters, C-4
 ladder representation, 11-25, 11-28
 operation, 11-26
 valid addressing modes, C-4
 valid file types, C-4
 file indicator (#), 6-10
 file organization
 compared to the programming software's
 file organization, 6-3
 data files, 6-4
 program, 6-3
 program files, 6-4
 file types, C-1, F-1
 Fill File (FLL), 11-10
 entering parameters, 11-12
 entering the instruction, 11-12
 execution times, 11-10
 function code, 11-13

instruction parameters, C-5
 ladder representation, 11-10
 using, 11-12
 valid addressing modes, C-5
 valid file types, C-5
 First Instruction on Rung screen, 17-2
 FLL, Fill File, 11-10
 forcing inputs and outputs, 18-35
 forcing external input data file bits, 18-35
 guide to forcing, 18-37
 forcing external output circuits, 18-38
 guide to forcing, 18-39
 FRD, Convert from BCD, 11-3
 function codes, listing, B-13
 functional areas, HHP, 4-7
 data monitor, 4-13
 home, 4-7
 menu, 4-8
 mode, 4-10
 multi-point function, 4-15
 program monitor, 4-11

G

general editing keys, identifying, 4-4
 general specifications, A-2
 controller, A-2
 HHP, A-9
 GEO, Greater Than or Equal, 9-8
 getting started, 5-1
 entering and running the program, 5-4
 monitoring operation, 5-9
 preparing to enter a new program, 5-2
 what to do first, 5-1
 what to do next, 5-12
 Greater Than (GRT), 9-7
 AND GRT
 entering the instruction, 9-7
 execution times, 9-7
 function code, 9-7
 instruction parameters, C-5
 ladder representation, 9-7
 valid addressing modes, C-5
 valid file types, C-5
 LD GRT
 entering the instruction, 9-7
 execution times, 9-7
 function code, 9-7
 instruction parameters, C-5
 ladder representation, 9-7
 valid addressing modes, C-5

- valid file types, C-5
 - OR GRT
 - entering the instruction, 9-7
 - execution times, 9-7
 - function code, 9-7
 - instruction parameters, C-5
 - ladder representation, 9-7
 - valid addressing modes, C-5
 - valid file types, C-5
 - Greater Than or Equal (GEO), 9-8
 - AND GEO
 - entering the instruction, 9-8
 - execution times, 9-8
 - function code, 9-8
 - instruction parameters, C-5
 - ladder representation, 9-8
 - valid addressing modes, C-5
 - valid file types, C-5
 - LD GEO
 - entering the instruction, 9-8
 - execution times, 9-8
 - function code, 9-8
 - instruction parameters, C-5
 - ladder representation, 9-8
 - valid addressing modes, C-5
 - valid file types, C-5
 - OR GEO
 - entering the instruction, 9-8
 - execution times, 9-8
 - function code, 9-8
 - instruction parameters, C-5
 - ladder representation, 9-8
 - valid addressing modes, C-5
 - valid file types, C-5
 - grounding the controller, 2-1
 - grounding your analog cable, 2-20
 - GRT, Greater Than, 9-7
 - guide to forcing
 - external input data file bits, 18-37
 - external output circuits, 18-39
- ## H
- hardware, features, 1-2
 - heat protection, 1-12
 - HHP
 - about, 4-1
 - accessories and replacement parts, A-11
 - changing defaults, 4-17
 - language, 4-17
 - LCD display contrast, 4-18
 - connecting, 3-1
 - dimensions, A-9
 - features, 4-2
 - functional areas, 4-7
 - identifying errors, 20-3
 - keys you use, 4-4
 - accessing additional characters, 4-5
 - context sensitivity, 4-4
 - power-up sequence, 4-6
 - programming examples, 16-1
 - specifications, A-9
 - High-Speed Counter (HSC), 14-4
 - entering parameters, 14-4
 - entering the instruction, 14-6
 - execution times, 14-4
 - function code, 14-6
 - instruction parameters, C-5
 - ladder representation, 14-4
 - types of, 14-5
 - bidirectional counter, 14-9
 - bidirectional counter with reset and hold, 14-9
 - bidirectional counter with reset and hold with a quadrature encoder, 14-12
 - up counter, 14-7
 - up counter with reset and hold, 14-7
 - valid addressing modes, C-5
 - valid file types, C-5
 - what happens when going to RRUN, 14-23
 - high-speed counter instructions
 - about, 14-1
 - High-Speed Counter (HSC), 14-4
 - High-Speed Counter Interrupt Disable (HSD), 14-21
 - High-Speed Counter Interrupt Enable (HSE), 14-21
 - High-Speed Counter Load (HSL), 14-15
 - High-Speed Counter Reset (RES), 14-19
 - High-Speed Counter Reset Accumulator (RAC), 14-20
 - in the paper drilling machine application example, 14-28
 - overview, 14-2
 - counter data file elements, 14-2
 - using status bits, 14-2
 - Update High-Speed Counter Image Accumulator (OUT), 14-23
 - High-Speed Counter Interrupt Disable (HSD), 14-21
 - execution times, 14-21
 - function code, 14-22
 - instruction parameters, C-5
 - ladder representation, 14-21

- using HSD, 14–22
 - entering the instruction, 14–22
 - operation, 14–22
 - valid addressing modes, C–5
 - valid file types, C–5
 - High-Speed Counter Interrupt Enable (HSE), 14–21
 - execution times, 14–21
 - function code, 14–21
 - instruction parameters, C–5
 - ladder representation, 14–21
 - using HSE, 14–21
 - entering the instruction, 14–21
 - operation, 14–22
 - valid addressing modes, C–5
 - valid file types, C–5
 - High-Speed Counter Load (HSL), 14–15
 - entering parameters, 14–15
 - entering the instruction, 14–15
 - execution times, 14–15
 - function code, 14–16
 - instruction parameters, C–5
 - ladder representation, 14–15
 - operation, 14–16
 - valid addressing modes, C–5
 - valid file types, C–5
 - High-Speed Counter Reset (RES), 14–19
 - entering the instruction, 14–19
 - execution times, 14–19
 - function code, 14–19
 - instruction parameters, C–8
 - ladder representation, 14–19
 - operation, 14–19
 - valid file types, C–8
 - High-Speed Counter Reset Accumulator (RAC), 14–20
 - ladder representation, 14–20
 - entering parameters, 14–20
 - entering the instruction, 14–20
 - execution times, 14–20
 - function code, 14–20
 - instruction parameters, C–8
 - operation, 14–21
 - valid addressing modes, C–8
 - valid file types, C–8
 - high-speed counter, wiring, 14–5
 - high-speed counter applications, wire the controller for, 2–23
 - home
 - description, 4–7
 - how to complete tasks, 4–8
 - screen definition, 4–7
 - HSC, High-Speed Counter, 14–4
 - HSD, High-Speed Counter Interrupt Disable, 14–21
 - HSE, High-Speed Counter Interrupt Enable, 14–21
 - HSL, High-Speed Counter Load, 14–15
- I**
- identifying controller faults, 20–11
 - IIM, Immediate Input with Mask, 12–8
 - Immediate Input with Mask (IIM), 12–8
 - entering parameters, 12–8
 - entering the instruction, 12–8
 - execution times, 12–8
 - function code, 12–8
 - instruction parameters, C–5
 - ladder representation, 12–8
 - valid addressing modes, C–5
 - valid file types, C–5
 - Immediate Output with Mask (IOM), 12–9
 - entering parameters, 12–9
 - entering the instruction, 12–9
 - execution times, 12–9
 - function code, 12–9
 - instruction parameters, C–6
 - ladder representation, 12–9
 - valid addressing modes, C–6
 - valid file types, C–6
 - indexed addressing, 6–9, C–2
 - entering the # character, 6–9
 - example of, 6–9
 - specifying, 6–9
 - input branching, 6–13
 - Input Channel Filtering, 7–3
 - Input Channel Filters and Update Times, 7–2
 - input file (I), 6–4
 - input filter response time, setting, 18–12, 18–14, 18–15, 18–16
 - input specifications
 - controller, A–3
 - HHP, A–9
 - Input States on Power Down, 1–11
 - input voltage ranges
 - 1761-L10BWA, 2–8
 - 1761-L10BWB, 2–11
 - 1761-L16AWA, 2–6
 - 1761-L16BBB, 2–15
 - 1761-L16BWA, 2–9
 - 1761-L16BWB, 2–12
 - 1761-L20AWA-5A, 2–17

- 1761-L20BWA-5A, 2-18
- 1761-L20BWB-5A, 2-19
- 1761-L32AAA, 2-14
- 1761-L32AWA, 2-7
- 1761-L32BBB, 2-16
- 1761-L32BWA, 2-10
- 1761-L32BWB, 2-13
- analog controllers, 2-22
- installing
 - the memory module, 4-3
 - the micro controller, 1-1
- instruction execution times
 - listing, B-16
 - worksheet, B-21
- instruction function codes, listing, B-13
- instruction keys, identifying, 4-4
- instruction list programming
 - applying to schematics, 6-16
 - description, 6-15
 - examples, 16-1
 - input rungs, 16-1
 - output rungs, 16-4
 - optimized instruction list, 16-1
 - programming considerations, 16-8
- instruction memory usage
 - listing, B-16
 - worksheet, B-21
- INT, Interrupt Subroutine, 13-19
- integer file (N), 6-4
- interrupt latency, B-20
 - STI, 13-15
 - user, B-20
- interrupt priorities, 13-16
- Interrupt Subroutine (INT), 13-19
 - entering the instruction, 13-19
 - execution times, 13-19
 - function code, 13-19
 - instruction parameters, C-6
 - ladder representation, 13-19
 - valid addressing modes, C-6
 - valid file types, C-6
- IOM, Immediate Output with Mask, 12-9
- isolated link coupler, installing, 3-3

J

- JMP, Jump, 12-2
- JSR, Jump to Subroutine, 12-3
- Jump (JMP), 12-2
 - entering parameters, 12-2

- entering the instruction, 12-2
- execution times, 12-2
- function code, 12-2
- instruction parameters, C-6
- ladder representation, 12-2
- using, 12-2
- valid addressing modes, C-6
- valid file types, C-6

- Jump to Subroutine (JSR), 12-3
 - entering the instruction, 12-4
 - execution times, 12-3
 - function code, 12-4
 - instruction parameters, C-6
 - ladder representation, 12-3
 - nesting subroutine files, 12-4
 - using, 12-4
 - valid addressing modes, C-6
 - valid file types, C-6

K

- keys
 - accessing additional characters, 4-5
 - context sensitivity, 4-4
 - diagnostic/troubleshooting keys, 4-4
 - general editing keys, 4-4
 - instruction keys, 4-4
 - navigation keys, 4-4
 - short-cut keys
 - for monitoring data table files, 18-27
 - for monitoring program files, 18-26
 - for selecting the language, 4-18
- keys you use, HHP, 4-4

L

- Label (LBL), 12-2
 - entering parameters, 12-2
 - entering the instruction, 12-3
 - execution times, 12-2
 - function code, 12-3
 - instruction parameters, C-6
 - ladder representation, 12-2
 - using, 12-2
 - valid addressing modes, C-6
 - valid file types, C-6
- LBL, Label, 12-2
- LCD display, changing the contrast, 4-18
- LD, Load, 8-3
- LDI, Load Inverted, 8-4
- LDT, Load True, 8-6

- LEDs, 20–1
 - error with controller, 20–2
 - normal controller operation, 20–1
- LEQ, Less Than or Equal, 9–6
- LES, Less Than, 9–5
- Less Than (LES), 9–5
 - AND LES
 - entering the instruction, 9–5
 - execution times, 9–5
 - function code, 9–5
 - instruction parameters, C–6
 - ladder representation, 9–5
 - valid addressing modes, C–6
 - valid file types, C–6
 - LD LES
 - entering the instruction, 9–5
 - execution times, 9–5
 - function code, 9–5
 - instruction parameters, C–6
 - ladder representation, 9–5
 - valid addressing modes, C–6
 - valid file types, C–6
 - OR LES
 - entering the instruction, 9–5
 - execution times, 9–5
 - function code, 9–5
 - instruction parameters, C–6
 - ladder representation, 9–5
 - valid addressing modes, C–6
 - valid file types, C–6
- Less Than or Equal (LEQ), 9–6
 - AND LEQ
 - entering the instruction, 9–6
 - execution times, 9–6
 - function code, 9–6
 - instruction parameters, C–6
 - ladder representation, 9–6
 - valid addressing modes, C–6
 - valid file types, C–6
 - LD LEQ
 - entering the instruction, 9–6
 - execution times, 9–6
 - function code, 9–6
 - instruction parameters, C–6
 - ladder representation, 9–6
 - valid addressing modes, C–6
 - valid file types, C–6
 - OR LEQ
 - entering the instruction, 9–6
 - execution times, 9–6
 - function code, 9–6
 - instruction parameters, C–6
 - ladder representation, 9–6
 - valid addressing modes, C–6
 - valid file types, C–6
- LFL, LIFO Load, 11–28
- LFU, LIFO Unload, 11–28
- LIFO Load (LFL), 11–28
 - entering the instruction, 11–28
 - execution times, 11–28
 - function code, 11–28
 - instruction parameters, C–6
 - ladder representation, 11–29
 - operation, 11–29
 - valid addressing modes, C–6
 - valid file types, C–6
- LIFO Unload (LFU), 11–28
 - entering the instruction, 11–28
 - execution times, 11–28
 - function code, 11–29
 - instruction parameters, C–6
 - ladder representation, 11–29
 - operation, 11–29
 - valid addressing modes, C–6
 - valid file types, C–6
- LIM, Limit Test, 9–10
- Limit Test (LIM), 9–10
 - AND LIM
 - entering parameters, 9–10
 - entering the instruction, 9–10
 - execution times, 9–10
 - function code, 9–10
 - instruction parameters, C–7
 - ladder representation, 9–10
 - valid addressing modes, C–7
 - valid file types, C–7
 - LD LIM
 - entering parameters, 9–10
 - entering the instruction, 9–10
 - execution times, 9–10
 - function code, 9–10
 - instruction parameters, C–7
 - ladder representation, 9–10
 - valid addressing modes, C–7
 - valid file types, C–7
 - OR LIM
 - entering parameters, 9–10
 - entering the instruction, 9–10
 - execution times, 9–10
 - function code, 9–10
 - instruction parameters, C–7
 - ladder representation, 9–10
 - valid addressing modes, C–7
 - valid file types, C–7

- Load (LD), 8–3
 - entering the instruction, 8–3
 - execution times, 8–3
 - instruction parameters, C–6
 - ladder representation, 8–3
 - using, 8–3
 - valid addressing modes, C–6
 - valid file types, C–6
 - Load Inverted (LDI), 8–4
 - entering the instruction, 8–5
 - execution times, 8–4
 - instruction parameters, C–6
 - ladder representation, 8–4
 - using, 8–5
 - valid addressing modes, C–6
 - valid file types, C–6
 - Load True (LDT), 8–6
 - entering the instruction, 8–6
 - execution times, 8–6
 - function code, 8–6
 - ladder representation, 8–6
 - using, 8–6
 - loading programs from a memory module, 19–2
 - lock program function, 18–17
 - setting, 18–17
 - logical addresses, using mnemonics to specify, 6–8
 - Low Voltage Directive, 1–1
- M**
- machine control, principles of, 6–1
 - manuals, related, P–5
 - Masked Comparison for Equal (MEQ), 9–9
 - AND MEQ
 - entering parameters, 9–9
 - entering the instruction, 9–9
 - execution times, 9–9
 - function code, 9–9
 - instruction parameters, C–7
 - ladder representation, 9–9
 - valid addressing modes, C–7
 - valid file types, C–7
 - LD MEQ
 - entering parameters, 9–9
 - entering the instruction, 9–9
 - execution times, 9–9
 - function code, 9–9
 - instruction parameters, C–7
 - ladder representation, 9–9
 - valid addressing modes, C–7
 - OR MEQ
 - entering parameters, 9–9
 - entering the instruction, 9–9
 - execution times, 9–9
 - function code, 9–9
 - instruction parameters, C–7
 - ladder representation, 9–9
 - valid addressing modes, C–7
 - valid file types, C–7
 - Masked Move (MVM), 11–16
 - entering parameters, 11–16
 - entering the instruction, 11–16
 - execution times, 11–16
 - function code, 11–16
 - instruction parameters, C–7
 - ladder representation, 11–16
 - operation, 11–17
 - updates to arithmetic status bits, 11–16
 - valid addressing modes, C–7
 - valid file types, C–7
 - Master Control Relay, 1–3
 - Master Control Reset (MCR), 12–6
 - entering the instruction, 12–6
 - execution times, 12–6
 - function code, 12–6
 - instruction parameters, C–7
 - ladder representation, 12–6
 - valid addressing modes, C–7
 - valid file types, C–7
 - master password, 18–2
 - master/sender communication, 15–1
 - math instructions
 - 32-bit addition and subtraction, 10–6
 - about, 10–1
 - Add (ADD), 10–4
 - Clear (CLR), 10–11
 - Divide (DIV), 10–9
 - Double Divide (DDV), 10–10
 - in the paper drilling machine application
 - example, 10–15
 - Multiply (MUL), 10–8
 - overview, 10–2
 - changes to the math register, S13 and S14, 10–3
 - entering the instructions, 10–2
 - overflow trap bit, S5/0, 10–2
 - updates to arithmetic status bits, 10–2
 - using indexed word addresses, 10–2
 - Scale Data (SCL), 10–12
 - Square Root (SQR), 10–11
 - Subtract (SUB), 10–5
 - using arithmetic status bits, 11–9

- MCR, Master Control Reset, 12–6
- memory module
 - clearing programs, 19–5
 - installation, 4–3
 - loading programs, 19–2
 - storing programs, 19–3
 - using, 19–1
- Memory Pop (MPP), 8–10
 - entering the instruction, 8–12
 - execution times, 8–10
 - function code, 8–12
 - ladder representation, 8–10
 - using, 8–11
- Memory Push (MPS), 8–10
 - entering the instruction, 8–10
 - execution times, 8–10
 - function code, 8–10
 - ladder representation, 8–10
 - using, 8–10
- Memory Read (MRD), 8–10
 - entering the instruction, 8–11
 - execution times, 8–10
 - function code, 8–11
 - ladder representation, 8–10
 - using, 8–11
- memory usage
 - listing, B–16
 - worksheet, B–21
- menu
 - description, 4–8
 - how to complete tasks, 4–9
 - screen definition, 4–9
- MEQ, Masked Comparison for Equal, 9–9
- Message (MSG), 15–1
 - application examples, 15–12
 - control block layout, 15–3
 - entering parameters, 15–2
 - entering the instruction, 15–6
 - error codes, 15–11
 - execution times, 15–2
 - function code, 15–7
 - instructional parameters, C–7
 - ladder representation, 15–2
 - timing diagram, 15–9
 - using status bits, 15–4
 - valid addressing modes, C–7
 - valid file types, C–7
- mnemonics, using
 - in instruction list programming, 6–15
 - in logical addresses, 6–8
- mode
 - changing editing modes, 17–3
 - changing remote controller modes, 18–23
 - description, 4–10
 - how to complete tasks, 4–11
 - screen definition, 4–10
- model for developing a logic program, 6–17
- modems
 - dial-up phone , D–7
 - leased-line, D–8
 - line drivers, D–8
 - radio, D–8
 - using with MicroLogix controllers, D–7
- modes of operation, 18–22
- monitoring
 - data table files, 18–26
 - changing the radix, 18–30
 - operation, 5–9
 - fault recovery procedure, 20–11
 - of the controller, 18–25
 - of the data, 5–10
 - of the program, 5–9
 - program files, 18–25
- motor starters (bulletin 509), surge suppressors, 1–9
- motor starters (bulletin 709), surge suppressors, 1–9
- mounting template, controller, A–7
- mounting the controller
 - using a DIN rail, 1–13
 - using mounting screws, 1–14
- MOV, Move, 11–15
- Move (MOV), 11–15
 - entering parameters, 11–15
 - entering the instruction, 11–15
 - execution times, 11–15
 - function code, 11–15
 - instructional parameters, C–7
 - ladder representation, 11–15
 - updates to arithmetic status bits, 11–15
 - valid addressing modes, C–7
 - valid file types, C–7
- move and logical instructions
 - And (AND), 11–18
 - Exclusive Or (XOR), 11–20
 - Masked Move (MVM), 11–16
 - Move (MOV), 11–15
 - Negate (NEG), 11–22
 - Not (NOT), 11–21
 - Or (OR), 11–19
 - overview, 11–13
 - changes to the math register, S13 and S14, 11–14

- entering parameters, 11–13
 - entering the instructions, 11–13
 - overflow trap bit, S5/0, 11–14
 - updates to arithmetic status bits, 11–14
 - using indexed word addresses, 11–14
- MPP, Memory Pop, 8–10
- MPS, Memory Push, 8–10
- MRD, Memory Read, 8–10
- MSG, Message, 15–1
- MUL, Multiply, 10–8
- multi-point function
- description, 4–15
 - how to complete tasks, 4–16
 - screen definition, 4–15
 - using, 18–31
 - automatically entering addresses, 18–32
 - changing multi-point addresses, 18–33
 - manually entering addresses, 18–31
 - removing multi-point addresses, 18–34
- Multiply (MUL), 10–8
- changes to the math register, 10–8
 - entering the instruction, 10–8
 - execution times, 10–8
 - function code, 10–8
 - instruction parameters, C–7
 - ladder representation, 10–8
 - updates to arithmetic status bits, 10–8
 - valid addressing modes, C–7
 - valid file types, C–7
- MVM, Masked Move, 11–16
- ## N
- navigation keys, identifying, 4–4
- NEG, Negate, 11–22
- Negate (NEG), 11–22
- entering the instruction, 11–22
 - execution times, 11–22
 - function code, 11–22
 - instruction parameters, C–7
 - ladder representation, 11–22
 - updates to arithmetic status bits, 11–22
 - valid addressing modes, C–7
 - valid file types, C–7
- NEQ, Not Equal, 9–4
- nested branching, 6–14
- new rung, starting, 17–2
- node address (S:15L), B–11
- Not (NOT), 11–21
- entering the instruction, 11–21
 - execution times, 11–21
 - function code, 11–21
 - instruction parameters, C–8
 - ladder representation, 11–21
 - updates to arithmetic status bits, 11–21
 - valid addressing modes, C–8
 - valid file types, C–8
- Not Equal (NEQ), 9–4
- AND NEQ
- entering the instruction, 9–4
 - execution times, 9–4
 - function code, 9–4
 - instruction parameters, C–8
 - ladder representation, 9–4
 - valid addressing modes, C–8
 - valid file types, C–8
- LD NEQ
- entering the instruction, 9–4
 - execution times, 9–4
 - function code, 9–4
 - instruction parameters, C–8
 - ladder representation, 9–4
 - valid addressing modes, C–8
 - valid file types, C–8
- OR NEQ
- entering the instruction, 9–4
 - execution times, 9–4
 - function code, 9–4
 - instruction parameters, C–8
 - ladder representation, 9–4
 - valid addressing modes, C–8
 - valid file types, C–8
- NOT, Not, 11–21
- number systems, 6–10
- radixes used, 6–10
- numeric constants, 6–10
- ## O
- One-Shot Rising (OSR), 8–7
- AND OSR
- entering parameters, 8–7
 - entering the instruction, 8–7
 - execution times, 8–7
 - function code, 8–7
 - instruction parameters, C–8
 - ladder representation, 8–7
 - valid addressing modes, C–8
 - valid file types, C–8
- ladder representation, 8–7
- LD OSR
- entering parameters, 8–7
 - entering the instruction, 8–7

- execution times, 8–7
 - function code, 8–7
 - instruction parameters, C–8
 - ladder representation, 8–7
 - valid addressing modes, C–8
 - valid file types, C–8
- operating cycle, controller's, 6–2
- Or (OR)
 - bit input instruction, 8–3
 - entering the instruction, 8–4
 - execution times, 8–3
 - instruction parameters, C–8
 - ladder representation, 8–3
 - using, 8–4
 - valid addressing modes, C–8
 - valid file types, C–8
 - word output instruction, 11–19
 - entering the instruction, 11–19
 - execution times, 11–19
 - function code, 11–19
 - instruction parameters, C–8
 - ladder representation, 11–19
 - updates to arithmetic status bits, 11–19
 - valid addressing modes, C–8
 - valid file types, C–8
- Or Block (ORB), 8–12
 - entering the instruction, 8–13
 - execution times, 8–12
 - ladder representation, 8–12
 - using, 8–13
- Or Inverted (ORI), 8–4
 - entering the instruction, 8–5
 - execution times, 8–4
 - instruction parameters, C–8
 - ladder representation, 8–4
 - using, 8–5
 - valid addressing modes, C–8
 - valid file types, C–8
- Or True (ORT), 8–6
 - entering the instruction, 8–6
 - execution times, 8–6
 - function code, 8–6
 - ladder representation, 8–6
 - using, 8–6
- OR, Or
 - bit input instruction, 8–3
 - word output instruction, 11–19
- ORB, Or Block, 8–12
- ORI, Or Inverted, 8–4
- ORT, Or True, 8–6
- OSR, One-Shot Rising, 8–7
- OUT, Output, 8–8
- OUT, Update High-Speed Counter Image Accumulator, 14–23
- Output (OUT), 8–8
 - entering the instruction, 8–8
 - execution times, 8–8
 - instruction parameters, C–8
 - ladder representation, 8–8
 - update high-speed counter image accumulator, 14–23
 - valid addressing modes, C–8
 - valid file types, C–8
- output branching, 6–13
- output contact protection, selecting, 1–7
- output file (O), 6–4
- output specifications, controller, A–4
- output voltage ranges
 - 1761-L10BWA, 2–8
 - 1761-L10BWB, 2–11
 - 1761-L16AWA, 2–6
 - 1761-L16BBB, 2–15
 - 1761-L16BWA, 2–9
 - 1761-L16BWB, 2–12
 - 1761-L20AWA-5A, 2–17
 - 1761-L20BWA-5A, 2–18
 - 1761-L20BWB-5A, 2–19
 - 1761-L32AAA, 2–14
 - 1761-L32AWA, 2–7
 - 1761-L32BBB, 2–16
 - 1761-L32BWA, 2–10
 - 1761-L32BWB, 2–13
- analog controllers, 2–22
- overflow trap bit, S5/0, 10–2
- overview
 - bit instructions, 8–3
 - branch instructions, 8–9
 - comparison instructions, 9–2
 - counter instructions, 8–21
 - FIFO and LIFO instructions, 11–23
 - high-speed counter instructions, 14–2
 - math instructions, 10–2
 - move and logical instructions, 11–13
 - Selectable Timed Interrupt (STI) function, 13–15
 - timer instructions, 8–14
- overwrite mode, 17–4
 - overwriting an instruction, 17–5

- overwriting an instruction's parameters, 17-4
 - ownership timeout, D-6
- P**
- parallel logic (OR), 6-12
 - parts, replacement, A-11
 - password
 - master password, 18-2
 - password override, 18-3
 - user password, 18-2
 - placing the controller in program mode, 5-2
 - planning considerations for a network, D-12
 - Power Considerations
 - Input States on Power Down, 1-11
 - Isolation Transformers, 1-10
 - Loss of Power Source, 1-11
 - other line conditions, 1-11
 - overview, 1-10
 - Power Distribution, 1-10
 - power-up sequence, 4-6
 - preparing to enter a new program, 5-2
 - clearing the current program, 5-2
 - placing the controller in program mode, 5-2
 - preventing excessive heat, 1-12
 - principles of machine control, 6-1
 - program
 - changing the configuration defaults, 18-1
 - clearing
 - from a memory module, 19-5
 - from the micro controller, 5-2, 19-6
 - organization of, 6-3
 - data files, 6-4
 - program files, 6-4
 - preparing to enter a new program, 5-2
 - retrieving from a memory module, 19-2
 - storing and accessing
 - normal operation, 6-6
 - power down, 6-6
 - power up, 6-7
 - saving, 6-5
 - storing to a memory module, 19-3
 - program constants, 6-10
 - program defaults, 18-1
 - program development model, 6-17
 - program faults, determining, 20-1, 20-5
 - program files, organization, 6-4
 - program flow control instructions
 - about, 12-1
 - Immediate Input with Mask (IIM), 12-8
 - Immediate Output with Mask (IOM), 12-9
 - in the paper drilling machine application
 - example, 12-10
 - Jump (JMP), 12-2
 - Jump to Subroutine (JSR), 12-3
 - Label (LBL), 12-2
 - Master Control Reset (MCR), 12-6
 - Return (RET), 12-3
 - Subroutine (SBR), 12-3
 - Suspend (SUS), 12-7
 - Temporary End (TND), 12-7
 - program logic
 - applying to your schematics, 6-11
 - instruction list programming, 6-16
 - ladder logic, 6-16
 - developing your logic program, 6-17
 - instruction list, 6-15
 - ladder logic basics
 - connecting blocks of logic, 6-14
 - input branching, 6-13
 - nested branching, 6-14
 - output branching, 6-13
 - parallel connections, 6-12
 - series connections, 6-12
 - program mode, changing to, 5-2
 - program monitor
 - description, 4-11
 - entering, 17-1
 - how to complete tasks, 4-12
 - screen definitions, 17-1
 - End of File screen, 17-2
 - First Instruction on Rung screen, 17-2
 - Start of File screen, 17-1
 - Start of Rung screen, 17-2
 - typical bit instruction screen, 4-11
 - program name, entering, 18-2
 - programming, considerations, 16-8
 - programming device, features, 4-2
 - programming overview, 6-1
 - protection methods for contacts, 1-7
 - publications, related, P-5
- Q**
- quadrature encoder input, 14-12

R

- RAC, High-Speed Counter Reset Accumulator, 14–20
 - radix, changing, 18–30
 - RC network, example, 1–8
 - recovering your work, 20–16
 - related publications, P–5
 - relay contact rating table, A–4
 - relays, surge suppressors for, 1–9
 - remote modes
 - program mode, 18–22
 - run mode, 18–22
 - test mode, 18–22
 - remote packet support, D–17
 - removing a program from a memory module, 19–5
 - replacement parts and accessories, A–11
 - RES, High-Speed Counter Reset, 14–19
 - RES, Reset, 8–27
 - Reset (RES)
 - high-speed counter reset, 14–19
 - entering the instruction, 14–19
 - execution times, 14–19
 - function code, 14–19
 - instruction parameters, C–8
 - ladder representation, 14–19
 - operation, 14–19
 - valid addressing modes, C–8
 - valid file types, C–8
 - timer/counter reset, 8–27
 - entering the instruction, 8–27
 - execution times, 8–27
 - function code, 8–27
 - instruction parameters, C–8
 - ladder representation, 8–27
 - valid addressing modes, C–8
 - valid file types, C–8
 - Reset (RST), 8–8
 - entering the instruction, 8–9
 - execution times, 8–8
 - instruction parameters, C–8
 - ladder representation, 8–8
 - using, 8–9
 - valid addressing modes, C–8
 - valid file types, C–8
 - resetting the high-speed counter accumulator, 14–19
 - response times, A–6
 - RET, Return, 12–3
 - Retentive Timer (RTO), 8–20
 - entering the instruction, 8–20
 - execution times, 8–20
 - function code, 8–20
 - instruction parameters, C–8
 - ladder representation, 8–20
 - using status bits, 8–20
 - valid addressing modes, C–8
 - valid file types, C–8
 - retrieving programs from a memory module, 19–2
 - Return (RET), 12–3
 - entering the instruction, 12–5
 - execution times, 12–3
 - function code, 12–5
 - instruction parameters, C–8
 - ladder representation, 12–3
 - nesting subroutine files, 12–4
 - using, 12–5
 - valid addressing modes, C–8
 - valid file types, C–8
 - RS-232 communication interface, D–1
 - RST, Reset, 8–8
 - RTO, Retentive Timer, 8–20
 - run always bit, setting, 18–5
- S**
- Safety Considerations
 - Disconnecting Main Power, 1–9
 - overview, 1–9
 - Periodic Tests of Master Control Relay Circuit, 1–10
 - Power Distribution, 1–10
 - Safety Circuits, 1–10
 - SBR, Subroutine, 12–3
 - Scale Data (SCL), 10–12
 - application example, 10–14
 - entering parameters, 10–12
 - entering the instruction, 10–13
 - execution times, 10–12
 - function code, 10–13
 - instruction parameters, C–9
 - ladder representative, 10–12, 10–14
 - updates to arithmetic status bits, 10–13
 - valid addressing modes, C–9
 - valid file types, C–9
 - SCL, Scale Data, 10–12
 - screen definitions
 - data monitor, 4–13
 - End of File, 17–2

- First Instruction on Rung, 17–2
 - home, 4–7
 - menu, 4–9
 - mode, 4–10
 - multi-point function, 4–15
 - program monitor, 4–11
 - Start of File, 17–1
 - Start of Rung, 17–2
- searching for specific addresses, 17–8
 - addresses that are displayed, 17–8
 - addresses you enter, 17–8
 - bit versus word addresses, 17–9
- Selectable Timed Disable (STD), 13–17
 - entering the instruction, 13–17
 - example, 13–17
 - execution times, 13–17
 - function code, 13–17
 - instruction parameters, C–9
 - ladder representation, 13–17
 - using, 13–17
 - valid addressing modes, C–9
 - valid file types, C–9
- Selectable Timed Enable (STE), 13–17
 - entering the instruction, 13–17
 - example, 13–17
 - execution times, 13–17
 - function code, 13–17
 - instruction parameters, C–9
 - ladder representation, 13–17
 - using, 13–17
 - valid addressing modes, C–9
 - valid file types, C–9
- Selectable Timed Interrupt (STI) function
 - basic programming procedure, 13–15
 - Interrupt Subroutine (INT), 13–19
 - operation, 13–15
 - interrupt latency and interrupt occurrences, 13–16
 - interrupt priorities, 13–16
 - status file data saved, 13–16
 - subroutine content, 13–15
 - overview, 13–15
 - Selectable Timed Disable (STD), 13–17
 - Selectable Timed Enable (STE), 13–17
 - Selectable Timed Start (STS), 13–18
 - STD/STE zone example, 13–17
- Selectable Timed Start (STS), 13–18
 - entering the instruction, 13–19
 - execution times, 13–18
 - function code, 13–19
 - instruction parameters, C–10
 - ladder representative, 13–18
 - valid addressing modes, C–10
 - valid file types, C–10
- Selecting Surge Suppressors, 1–7
- selecting the language
 - using short-cut keys, 4–18
 - using the menu option, 4–17
- Sequencer Compare (SQC), 13–6
 - entering parameters, 13–6
 - entering the instruction, 13–10
 - execution times, 13–6
 - function code, 13–10
 - instruction parameters, C–9
 - ladder representation, 13–6
 - operation, 13–11
 - using, 13–10
 - valid addressing modes, C–9
 - valid file types, C–9
- sequencer instructions
 - overview, 13–6
 - effects on index register S:24, 13–6
 - entering the instructions, 13–6
 - Sequencer Compare (SQC), 13–6
 - Sequencer Load (SQL), 13–12
 - Sequencer Output (SQO), 13–6
- Sequencer Load (SQL), 13–12
 - entering parameters, 13–12
 - entering the instruction, 13–13
 - execution times, 13–12
 - function code, 13–13
 - instruction parameters, C–9
 - ladder representation, 13–12
 - operation, 13–14
 - valid addressing modes, C–9
 - valid file types, C–9
- Sequencer Output (SQO), 13–6
 - entering parameters, 13–6
 - entering the instruction, 13–8
 - execution times, 13–6
 - function code, 13–8
 - instruction parameters, C–9
 - ladder representation, 13–6
 - operation, 13–9
 - using, 13–7
 - valid addressing modes, C–9
 - valid file types, C–9
- series logic (AND), 6–12
- Set (SET), 8–8
 - entering the instruction, 8–9
 - execution times, 8–8
 - instruction parameters, C–9
 - ladder representation, 8–8
 - using, 8–9
 - valid addressing modes, C–9

- valid file types, C-9
 - SET, Set, 8-8
 - short-cut keys
 - for monitoring data table files, 18-27
 - for monitoring program files, 18-26
 - for selecting the language, 4-18
 - single scan (SSN) mode, 18-22
 - sinking and sourcing circuits, overview, 2-2
 - slave/receiver communication, 15-2
 - spacing the controller, 1-12
 - specifications
 - general, A-2
 - controller, A-2
 - HHP, A-9
 - input
 - controller, A-3
 - HHP, A-9
 - output, controller, A-4
 - response times, A-6
 - SQC, Sequencer Compare, 13-6
 - SQL, Sequencer Load, 13-12
 - SQO, Sequencer Output, 13-6
 - SQR, Square Root, 10-11
 - Square Root (SQR), 10-11
 - entering the instruction, 10-12
 - execution times, 10-11
 - function code, 10-12
 - instruction parameters, C-9
 - ladder representative, 10-11
 - updates to arithmetic status bits, 10-11
 - valid addressing modes, C-9
 - valid file types, C-9
 - Start of File screen, 17-1
 - Start of Rung (SOR) symbol, 17-2
 - Start of Rung screen, 17-2
 - start-up protection, setting, 18-6
 - status file
 - descriptions, B-2
 - overview, B-1
 - status file (S), 6-4
 - STD, Selectable Timed Disable, 13-17
 - STE, Selectable Timed Enable, 13-17
 - STI enabled bit, setting, 18-10
 - STI setpoint, setting, 18-9
 - storing programs
 - power down, 6-6
 - power up, 6-7
 - saving, 6-5
 - to a memory module, 19-3
 - STS, Selectable Timed Start, 13-18
 - SUB, Subtract, 10-5
 - Subroutine (SBR), 12-3
 - entering the instruction, 12-5
 - execution times, 12-3
 - function code, 12-5
 - instruction parameters, C-8
 - ladder representation, 12-3
 - nesting subroutine files, 12-4
 - using, 12-5
 - valid addressing modes, C-8
 - valid file types, C-8
 - Subtract (SUB), 10-5
 - entering the instruction, 10-5
 - execution times, 10-5
 - function code, 10-5
 - instruction parameters, C-10
 - ladder representation, 10-5
 - updates to arithmetic status bits, 10-5
 - valid addressing modes, C-10
 - valid file types, C-10
 - surge suppressors, 1-7
 - example, 1-8
 - for contactor, 1-9
 - for motor starters, 1-9
 - for relays, 1-9
 - recommended, 1-9
 - SUS, Suspend, 12-7
 - Suspend (SUS), 12-7
 - entering parameters, 12-7
 - entering the instruction, 12-7
 - execution times, 12-7
 - function code, 12-7
 - instruction parameters, C-10
 - ladder representation, 12-7
 - valid addressing modes, C-10
 - valid file types, C-10
 - system configuration, DH-485 connection
 - examples, D-14
 - system connection, 3-1
 - system considerations, 16-8
- ## T
- Temporary End (TND), 12-7
 - entering the instruction, 12-7
 - execution times, 12-7
 - function code, 12-7
 - instruction parameters, C-10
 - ladder representation, 12-7

- valid addressing modes, C-10
 - valid file types, C-10
 - test modes, 18-22
 - continuous scan (CSN), 18-22
 - single scan (SSN), 18-22
 - timer file (T), 6-4
 - timer instructions
 - overview
 - addressing structure, 8-15
 - entering parameters, 8-14
 - entering the instructions, 8-15
 - Retentive Timer (RTO), 8-20
 - Timer Off-Delay (TOF), 8-18
 - Timer On-Delay (TON), 8-16
 - Timer Off-Delay (TOF), 8-18
 - entering the instruction, 8-18
 - execution times, 8-18
 - function code, 8-18
 - instruction parameters, C-10
 - ladder representation, 8-18
 - using status bits, 8-18
 - valid addressing modes, C-10
 - valid file types, C-10
 - Timer On-Delay (TON), 8-16
 - entering the instruction, 8-16
 - execution times, 8-16
 - function code, 8-16
 - instruction parameters, C-10
 - ladder representation, 8-16
 - using status bits, 8-16
 - valid addressing modes, C-10
 - valid file types, C-10
 - timing diagram, message instruction, 15-9
 - TND, Temporary End, 12-7
 - TOD, Convert to BCD, 11-2
 - TOF, Timer Off-Delay, 8-18
 - TON, Timer On-Delay, 8-16
 - trace feature, using, 20-8
 - tracing specific addresses
 - addresses that are displayed, 20-9
 - addresses that you enter, 20-9
 - bit versus word addresses, 20-9
 - troubleshooting
 - automatically clearing faults, 20-12
 - contacting Allen-Bradley for assistance, P-6, 20-16
 - controller error recovery model, 20-10
 - determining controller faults, 20-1
 - identifying controller faults, 20-11
 - keys you use, 4-4
 - manually clearing faults, 20-11
 - power failure, 20-16
 - recovering your work, 20-16
 - understanding the controller LED status, 20-1
 - using the fault routine, 20-12
- ## U
- understanding
 - addressing, 6-7
 - file organization, 6-3
 - how programs are stored and accessed, 6-5
 - ladder logic programs, 6-11
 - the HHP keys' context sensitivity, 4-4
 - up counter
 - operation, 14-7
 - overview, 14-5
 - up counter with reset and hold
 - operation, 14-7
 - overview, 14-5
 - Update High-Speed Counter Image
 - Accumulator (OUT), 14-23
 - entering the instruction, 14-23
 - execution times, 14-23
 - ladder representation, 14-23
 - operation, 14-23
 - updating the high-speed counter accumulator, 14-23
 - user interrupt latency, B-20
 - user password, 18-2
 - using the trace feature, 20-8
- ## V
- valid addressing modes, C-2
 - varistors
 - example, 1-8
 - recommended, 1-8
 - viewing data table files, 18-28
- ## W
- watchdog scan, setting, 18-11
 - what to do first, 5-1
 - wire types, 2-3
 - wiring
 - analog controllers, 2-17
 - your controller for high-speed counter applications, 2-23

Index

MicroLogix™ 1000 with Hand-Held Programmer
(HHP) User Manual

wiring analog channels, 2–21

wiring diagrams

1761-L10BWA, 2–8

1761-L10BWB, 2–11

1761-L16AWA, 2–6

1761-L16BBB, 2–15

1761-L16BWA, 2–9

1761-L16BWB, 2–12

1761-L20AWA-5A, 2–17

1761-L20BWA-5A, 2–18

1761-L20BWB-5A, 2–19

1761-L32AAA, 2–14

1761-L32AWA, 2–7

1761-L32BBB, 2–16

1761-L32BWA, 2–10

1761-L32BWB, 2–13

wiring recommendations, 2–3

X

XOR, Exclusive Or, 11–20

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

Europe/Middle East/Africa: Rockwell Automation, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640

Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846