USER MANUAL

SPECTRUM CONTROLS

An Allient Company

User's Manual Pub. 0300217-06 Rev. A0

1769 4-Channel Isolated Analog HART Output Module

Catalog Number: 1769sc-OF4IH, 1769sc-OF4IHK

Important Notes

1. Please read all the information in this owner’s guide before installing the product.
2. The information in this owner's guide applies to hardware Series A and firmware version 1.00 or later.
3. This guide assumes that the reader has a full working knowledge of the relevant processor.

Notice

The products and services described in this owner's guide are useful in a wide variety of applications. Therefore, the user and others responsible for applying the products and services described herein are responsible for determining their acceptability for each application. While efforts have been made to provide accurate information within this owner's guide, Spectrum Controls, Inc. assumes no responsibility for the accuracy, completeness, or usefulness of the information herein.

Under no circumstances will Spectrum Controls, Inc. be responsible or liable for any damages or losses, including indirect or consequential damages or losses, arising out of either the use of any information within this owner's guide or the use of any product or service referenced herein.

No patent liability is assumed by Spectrum Controls, Inc. with respect to the use of any of the information, products, circuits, programming, or services referenced herein.

The information in this owner's guide is subject to change without notice.

Limited Warranty

Spectrum Controls, Inc. warrants that its products are free from defects in material and workmanship under normal use and service, as described in Spectrum Controls, Inc. literature covering this product, for a period of 1 year. The obligations of Spectrum Controls, Inc. under this warranty are limited to replacing or repairing, at its option, at its factory or facility, any product which shall, in the applicable period after shipment, be returned to the Spectrum Controls, Inc. facility, transportation charges prepaid, and which after examination is determined, to the satisfaction of Spectrum Controls, Inc., to be thus defective.

This warranty shall not apply to any such equipment which shall have been repaired or altered except by Spectrum Controls, Inc. or which shall have been subject to misuse, neglect, or accident. In no case, shall the liability of Spectrum Controls, Inc. exceed the purchase price. The aforementioned provisions do not extend the original warranty period of any product which has either been repaired or replaced by Spectrum Controls, Inc.

Table of Contents

Preface

NOTE	This is a re-issue of an existing manual, with some corrections, and updated Configuration information.

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

- Who should use this manual
- How to use this manual
- Related documentation
- Technical support
- Documentation
- Conventions used in this manual

Who Should Use This Manual

Use this manual if you are responsible for designing, installing, programming, or troubleshooting control systems that use Allen-Bradley Compact™ I/O and/or compatible controllers, such as MicroLogix 1500 or CompactLogix.

How to Use This Manual

As much as possible, we organized this manual to explain, in a task-by-task manner, how to install, configure, program, operate and troubleshoot a control system using the 1769sc-OF4IH.

Related Documentation

The table below provides a listing of publications that contain important information about Allen-Bradley PLC systems.

For	Refer to this Document	Allen-Bradley Pub. No.
User instructions	MicroLogix™ 1500 User Manual	1764-UM001A
Product information	1769 Compact Discrete Input/Output Modules Product Data	1769-2.1
Overview of MicroLogix 1500 system	MicroLogix™ 1500 System Overview	1764-SO001B
Overview of Compact IO system	Compact™ I/O System Overview	1769-SO001A
User Instructions	CompactLogix User Manual	1769-UM007B

For	Refer to this Document	Allen-Bradley Pub. No.
Wiring and grounding information	Allen-Bradley Programmable Controller Grounding and Wiring Guidelines	1770-4.1

Technical Support

For technical support, please contact your local Rockwell Automation TechConnect Office for all Spectrum products. Contact numbers are as follows:

- USA 1-440-646-6900 (US/global, English only
- United Kingdom +44 0 1908 635 230 (EU phone, UK local)
- Australia, China, India, and other East Asia locations: 1-800-722-778 or +61 39757 1502
- Mexico 001-888-365-8677
- Brazil 55-11-5189-9500 (general support)
- Europe +49-211-41553-630 (Germany/general support)

or send an email to support@spectrumcontrols.com

Documentation

If you would like a manual, you can download a free electronic version from the Internet at www.spectrumcontrols.com.

Conventions Used in This Manual

The following conventions are used throughout this manual:

- Bulleted lists (like this one) provide information not procedural steps.
- lists provide sequential steps or hierarchical information.
- *Italic* type is used for emphasis.
- **Bold** type identifies headings and sub-headings:

WARNING	Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. These messages help you to identify a hazard, avoid a hazard, and recognize the consequences.

ATTENTION	Actions ou situations risquant d’entraîner des blessures pouvant être mortelles, des dégâts matériels ou des pertes financières. Les messages « Attention » vous aident à identifier un danger, à éviter ce danger et en discerner les conséquences.

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

Chapter 1
Module Overview

This chapter describes the 1769sc-OF4IH and the 1769sc-OF4IHK isolated HART analog output modules and explains how the modules provide four isolated current outputs with HART communication. Other than the conformal coating, both modules are identical so all information applicable to the 1769sc-OF4IH also applies to the K version. The following section covers:

- The module's hardware and diagnostic features.
- An overview of the system and module operation.
- Compatibility.

Section 1.1 General Description

The OF4IH is a four-channel isolated module that allows each channel to be configured independently for either 0 to 20 mA or 4 to 20 mA with, or without, HART communication. The module converts the digital value stored in each channel's output command word (that is, output words 0 to 3) to an analog current signal. If HART is enabled on a specific channel, the user also has the ability to send and receive HART communication to, and from, the connected HART device. HART data is sent and received using the module's input and output image table. Over-range/under-range detection and indication is also provided by the module for each channel.

Section 1.2 Output Types and Ranges

The table below lists the output ranges for the module.

Table 1-1. Current Output Ranges

Current Output Range
0 to 20 mA
4 to 20 mA

Section 1.3 Data Formats

For each channel the data can be configured for:

- Engineering Units ×1
- Scaled-for-PID
- Percent of full scale
- Raw/proportional counts

Section 1.4 Hardware Features

The module contains a removable terminal block. Each channel has a dedicated ground which is isolated from the remaining channels by 500 VDC. differential inputs (that is, each channel will have a dedicated ground).

NOTE	Do not short the channel grounds together unless you plan to remove the channel-to-channel isolation.

Module configuration is normally done via the controller's programming software. In addition, some controllers support configuration via the user program. In either case, the module configuration is stored in the memory of the controller. Refer to your controller's user manual for more information.

1.4.1 General Diagnostic Features

The module contains a diagnostic LED that helps you identify the source of problems that may occur during power-up or during normal channel operation. The LED indicates both status and power. Power-up and channel diagnostics are explained in Chapter 8 Diagnostics and Troubleshooting

Section 1.5 System Overview

The modules communicate to the controller through the bus interface. The modules also receive 5 VDC and 24 VDC power through the bus interface.

1.5.1 Module Power-Up

At power-up, the module performs a check of its internal circuits, memory, and basic functions. During this time, the module status LED remains off. If no faults are found during power-up diagnostics, the module status LED is turned on.

After power-up checks are complete, the module waits for valid channel configuration data. If an invalid configuration is detected, the module generates a configuration error. Once a channel is properly configured and enabled, it continuously converts the output command value (that is, output words 0 to 3) to a proportional analog signal that is within the output range selected for that channel.

Each time a channel command value is read by the output module, that data value is tested by the module for an over-range or under-range condition. If such a condition is detected, a unique bit is set in the channel status word. The channel status word is described in Section 5.3 Input Data File.

Using the module image table, the controller reads the two's complement binary converted input data from the module. This typically occurs at the end of the program scan or when commanded by the control program. If the controller and the module determine that the data transfer has been made without error, the data is used in the control program.

1.5.2 Module Operation

When the module receives a new command value from the output image, the module's circuitry converts the digital value to an analog current signal using a DAC (Digital-to-Analog Converter). The resulting signal is made available for the specific channel via the removable terminal block. If HART is enabled on a channel, the HART data is acquired by means of an onboard HART modem.

NOTE	The HART data is sent and received asynchronously from the analog acquisition process and therefore does not directly affect the analog update time.

The module is designed to support up to 4 isolated channels which can be independently configured for current, or current with HART.

The HART data, if enabled, is converted directly to a block of twenty controller input tags. The data within this block of twenty tags is multiplexed. For information on HART and how to demultiplex the HART data, refer to Chapter 6.

See the block diagram below.

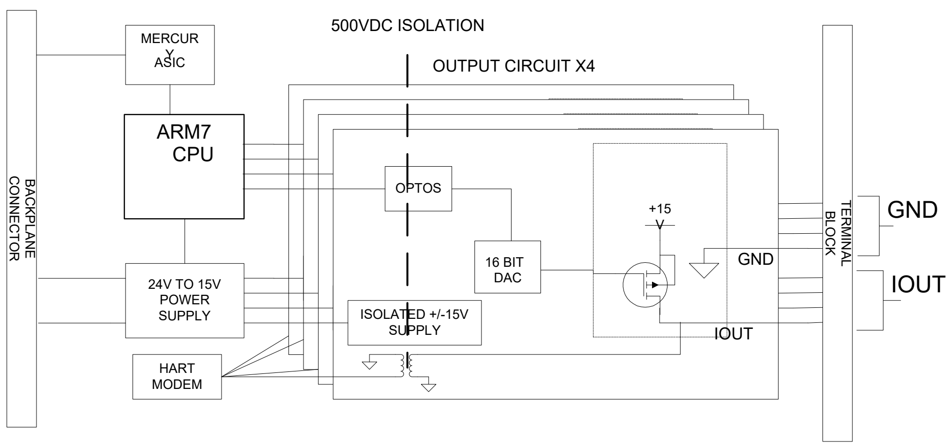


Figure 1-1. 1769sc-OF4IH Block Diagram

Chapter 2
Installation and Wiring

This chapter explains how to:

- Tools and Equipment
- Compliance to European Union directives
- Power requirements
- General considerations
- Mounting

Section 2.1 Required Tools and Equipment

You need the following tools and equipment:

- Medium blade or cross-head screwdriver
- Analog output device
- Shielded, twisted-pair cable for wiring (Belden™ 8761 or equivalent for current outputs)
- Controller (for example, a MicroLogix™ 1500 or CompactLogix™ controller)
- Programming device and software (for example, RSLogix 500™ or RSLogix 5000™)

Section 2.2 Compliance to European Union Directives

This product is approved for installation within the European Union and EEA regions. It has been designed and tested to meet the following directives.

2.2.1 EMC Directive

This product is approved for installation within the European Union and EEA regions. It has been designed and tested to meet the following directives.

The 1769sc-OF4IH module is tested to meet Council Directive 2014/30/EU Electromagnetic Compatibility (EMC) and the following standards, in whole or in part, documented in a technical construction file:

- EN 61131-2 Programable controllers, Part 2 - Equipment requirements and tests.
- EN 61000-6-2 Electromagnetic compatibility (EMC) – Part 6-2: Generic standards – Immunity standard for industrial environments.

- EN 61000-6-4 Electromagnetic compatibility (EMC) – Part 6-4: Generic standards – Emission standard for industrial environments.

UKCA Electromagnetic Compatibility Regulations 2016

- BS EN 61131-2, BS EN 61000-6-4, BS EN 61000-6-2

This product is intended for use in an industrial environment.

2.2.2 ATEX Directive

This product is tested to meet Council Directive 2014/30/U/ATEX, and the following standards, in whole or in part, documented in a technical construction file:

- EN 60079-0 Explosive atmospheres – Part 0: Equipment – General requirements.
- EN 60079-7 Explosive atmospheres – Part 7: Equipment protection by increased safety "e".

This module also meets the standards for the United Kingdom Equipment and Protective Systems Intended for use in Potentially Explosive Atmospheres Regulations 2016:

- BS EN 60079-0
- BS EN 60079-7

Section 2.3 Power Requirements

You must ensure that your power supply has sufficient current output to support your system configuration. The module receives power through the bus interface from the +5 VDC/+24 VDC system power supply. The maximum current drawn by the module is shown in the table below:

5 VDC	24 VDC
180 mA	200 mA

The system power supply may be a 1769-PA2, -PB2, -PA4, -PB4, or the internal supply of the MicroLogix 1500 packaged controller. The module cannot be located more than 8 modules away from the system power supply.

Section 2.4 Considerations

Compact I/O is suitable for use in an industrial environment when installed in accordance with these instructions. Specifically, this equipment is intended for use in clean, dry environments (Pollution degree 2[1] and to circuits not exceeding Over Voltage Category II[2] (IEC 60664-1)[3]

[1] Pollution Degree 2 is an environment where, normally, only non-conductive pollution occurs except that occasionally a temporary conductivity caused by condensation shall be expected.
[2] Over Voltage Category II is the load level section of the electrical distribution system. At this level, transient voltages are controlled and do not exceed the impulse voltage capability of the product's insulation.
[3] Pollution Degree 2 and Over Voltage Category II are International Electrotechnical Commission (IEC) designations.

2.4.1 Hazardous Location Considerations

This equipment is suitable for use in Class I, Division 2, Groups A, B, C, D or non-hazardous locations only. The following WARNING statement applies to use in hazardous locations.

WARNING	EXPLOSION HAZARD • Substitution of components may impair suitability for Class I, Division 2. Do not replace components or disconnect equipment unless power has been switched off or the area is known to be non-hazardous. • Do not connect or disconnect components unless power has been switched off or the area is known to be non-hazardous. • Device shall be installed in an enclosure which can only be opened with the use of a tool. • All wiring must comply with N.E.C. article 501-4(b), 502-4(b), or 503-3(b), as appropriate for Class I, Class II, and Class III equipment.

2.4.2 Prevent Electrostatic Discharge

WARNING	Electrostatic discharge can damage integrated circuits or semiconductors if you touch analog I/O module bus connector pins or the terminal block on the output module. Follow these guidelines when you handle the module: • Touch a grounded object to discharge static potential. • Wear an approved wrist-strap grounding device. • Do not touch the bus connector or connector pins. • Do not touch circuit components inside the module. • If available, use a static-safe workstation. • When it is not in use, keep the module in its static-shield bag.

2.4.3 Remove Power

WARNING	Remove power before removing or inserting this module. When you remove, or insert, a module with power applied, an electrical arc may occur. An electrical arc can cause personal injury or property damage by: • Sending an erroneous signal to your system's field devices, causing unintended machine motion. • Causing an explosion in a hazardous environment. • Causing an electrical arc. Such arcing causes excessive wear to contacts on both the module and its mating connector and may lead to premature failure.

2.4.4 Selecting a Location

Reducing Noise

Most applications require installation in an industrial enclosure to reduce the effects of electrical interference. Analog inputs are highly susceptible to electrical noise. Electrical noise coupled to the analog inputs will reduce the performance (accuracy) of the module.

Group your modules to minimize adverse effects from radiated electrical noise. Consider the following conditions when selecting a location for the analog module. Position the module:

- Away from sources of electrical noise such as hard-contact switches, relays, and AC motor drives.
- Away from high voltage conductors.

In addition, route shielded, twisted-pair analog output wiring away from any high voltage I/O wiring.

Reducing Heat

To avoid complications when operating in ambient temperatures above 55 °C, the following recommendations should be followed. Position the module:

- Away from heat sources such as transformers, variable frequency drives, and cabinet heaters.
- Avoid installing the module adjacent to modules which generate over 4 W of heat, such as the 1769-HSC. The table below lists modules which should be avoided if possible:

Table 2-1. Modules to Avoid

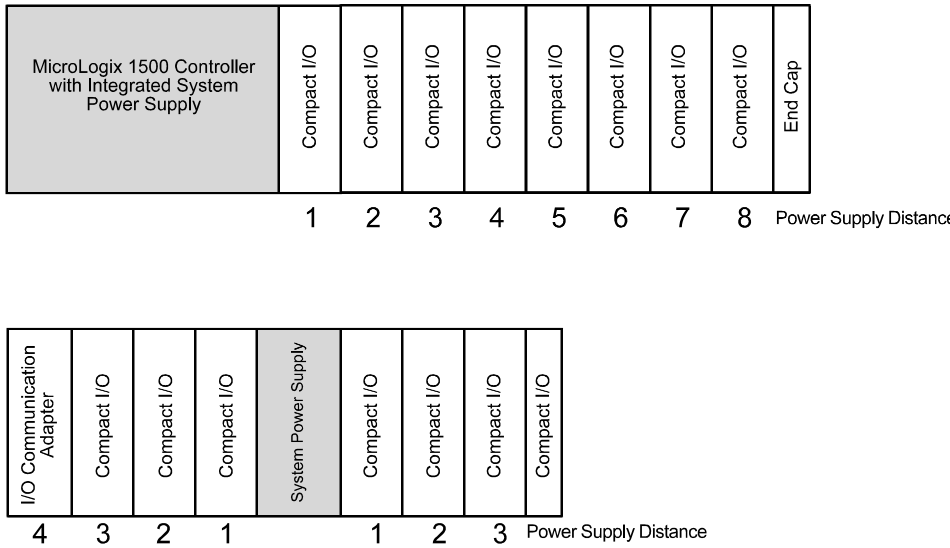
Module Catalog	Max Thermal Dissipation
1769-ADN	4.7
1769-HSC	6.2
1769-IQ32	4.8
1769-OW16	4.8
1769-OB32	4.5

NOTE	If the OF4IH is to be installed adjacent to one of the modules listed in the table above and the ambient temperature is above 55 °C, then derating of the module is required to avoid thermal shutdown. Assume the thermal dissipation of the OF4IH to be 2 W fixed, plus an additional 0.5 W per channel. For example, if the OF4IH is to be installed adjacent to the 1769-HSC, no more than 2 channels should be used on the OF4IH.

Power Supply Distance

You can install as many modules as your power supply can support. However, all 1769 I/O modules have a power supply distance rating. The maximum I/O module rating is 8, which means that a module may not be located more than 8 modules away from the system power supply.

Figure 2-1. Power Supply Distance



Section 2.5 Mounting

WARNING	Keeping module free of debris and avoiding overheating: • Do not remove protective debris strip until after the module and all other equipment near the module is mounted and the wiring is complete. • Once wiring is complete, and the module is free of debris, carefully remove protective strip. • Failure to remove strip before operating can cause overheating.

2.5.1 Minimum Spacing

Maintain spacing from enclosure walls, wire ways, adjacent equipment, etc. Allow 50.8 mm (2 in.) of space on all sides for adequate ventilation, as shown:

Figure 2-2. Minimum Spacing

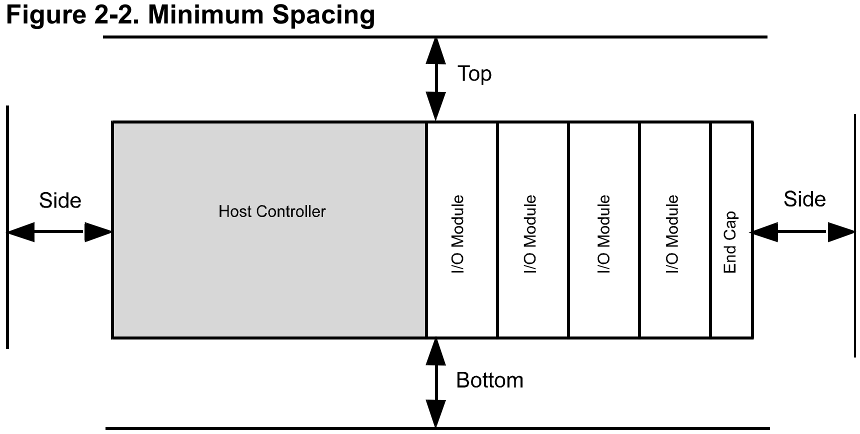
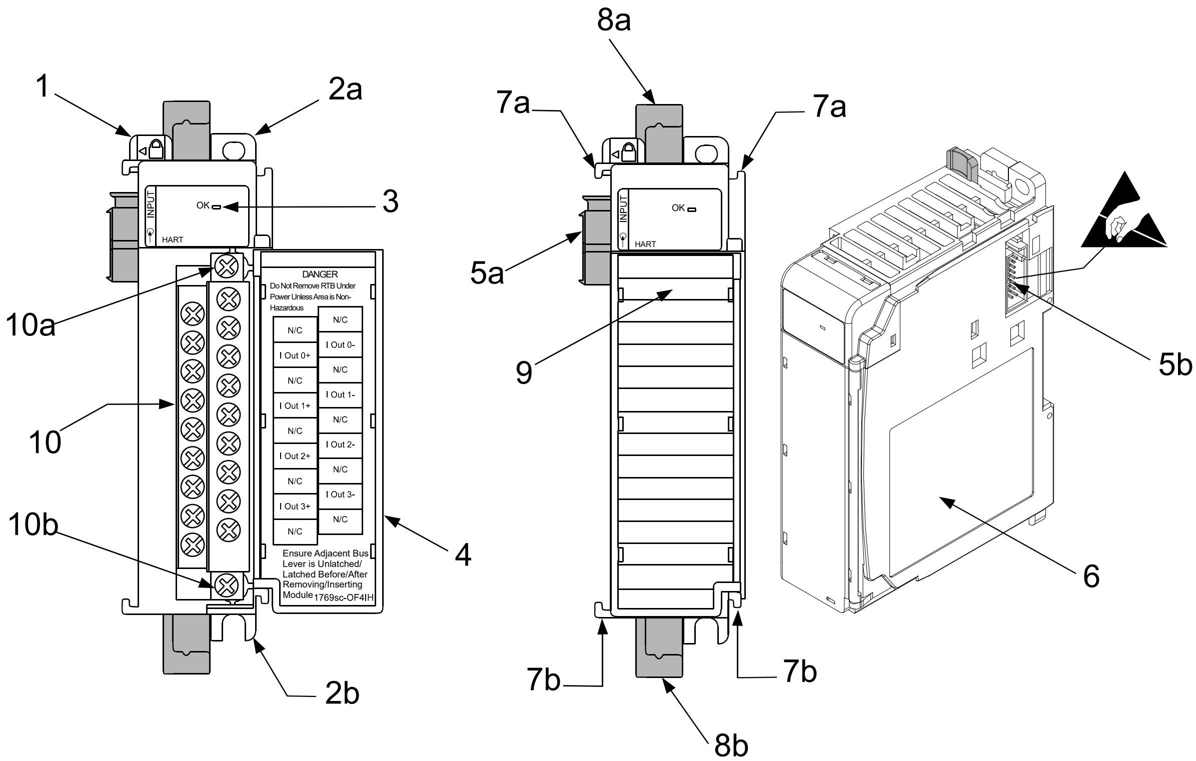


Figure 2-3. Module Parts List

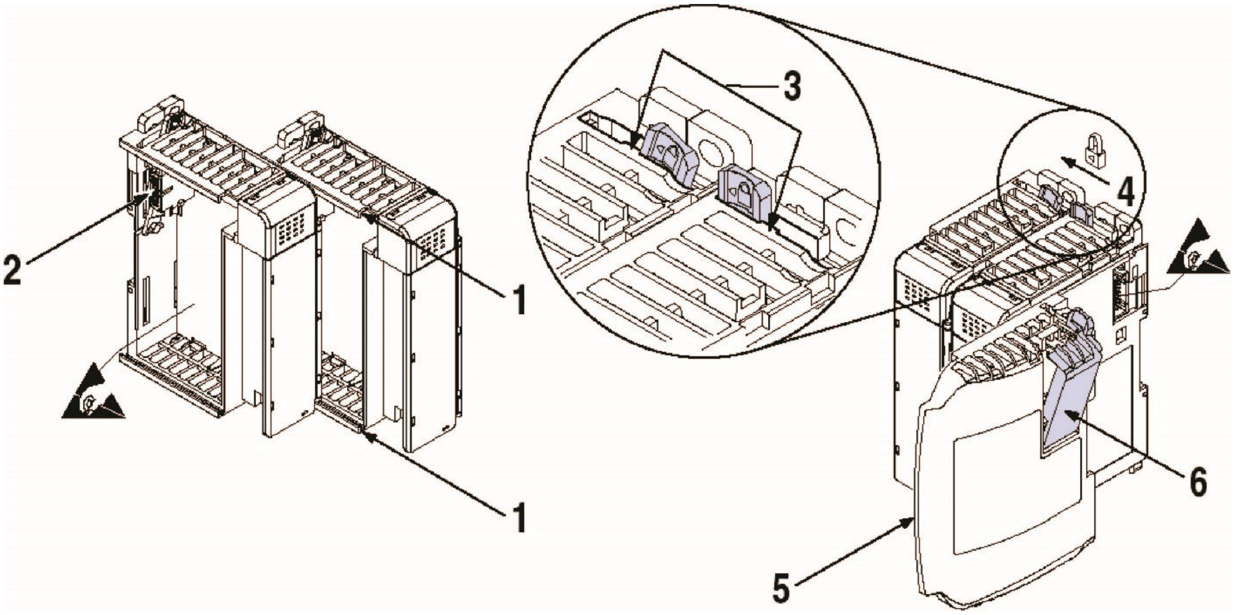


Item	Description
1	bus lever
2a	upper panel mounting tab
2b	lower panel mounting tab
3	module status LED
4	module door with terminal identification label
5a	movable bus connector (bus interface) with female pins
5b	stationary bus connector (bus interface) with male pins
6	nameplate label
7a	upper tongue-and-groove slots
7b	lower tongue-and-groove slots
8a	upper DIN rail latch
8b	lower DIN rail latch
9	write-on label for user identification tags
10	removable terminal block (RTB) with finger-safe cover
10a	RTB upper retaining screw
10b	RTB lower retaining screw

The module can be attached to the controller or an adjacent I/O module before or after mounting. For mounting instructions, see Panel Mounting Using the Dimensional Template, or DIN Rail Mounting. To work with a system that is already mounted, see Replacing a Single Module within a System.

The following procedure shows you how to assemble the Compact I/O system.

Figure 2-4. Module Assembly



1. Disconnect power.
2. Check that the bus lever of the module to be installed is in the unlocked (fully right) position.

NOTE	If the module is being installed to the left of an existing module, check that the right-side adjacent module's bus lever is in the unlocked (fully right) position.

3. Use the upper and lower tongue-and-groove slots (1) to secure the modules together (or to a controller).
4. Move the module back along the tongue-and-groove slots until the bus connectors (2) line up with each other.
5. Push the bus lever back slightly to clear the positioning tab (3). Use your fingers or a small screwdriver.
6. To allow communication between the controller and module, move the bus lever fully to the left (4) until it clicks. Ensure it is locked firmly in place.

WARNING	When attaching I/O modules, it is very important that the bus connectors are securely locked together to ensure proper electrical connection.

7. Attach an end cap terminator (5) to the last module in the system by using the tongue-and-groove slots as before.
8. Lock the end cap bus terminator (6).

WARNING	A 1769-ECR or 1769-ECL right or left end cap respectively must be used to terminate the end of the 1769 communication bus.

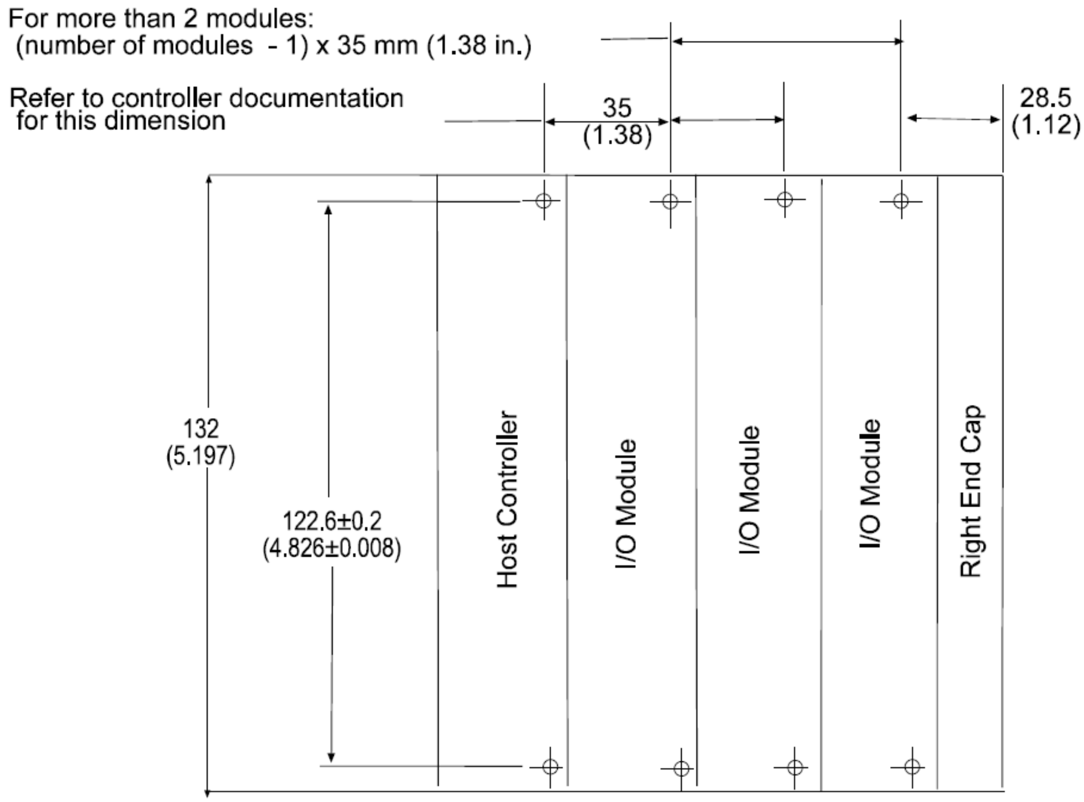
Section 2.6 Panel Mounting

WARNING	During panel or DIN rail mounting of all devices, be sure that all debris (metal chips, wire strands, etc.) is kept from falling into the module. Debris that falls into the module could cause damage at power up.

Mount the module to a panel using two screws per module. Use M4 or #8 pan head screws. Mounting screws are required on every module.

Panel Mounting Using the Dimensional Template

Figure 2-5. Dimensional Template



Panel Mounting Procedure Using Modules as a Template

The following procedure allows you to use the assembled modules as a template for drilling holes in the panel. If you have sophisticated panel mounting equipment, you can use the dimensional template provided on the previous page.

Due to module mounting hole tolerance, it is important to follow these procedures:

1. On a clean work surface, assemble no more than three modules.
2. Using the assembled modules as a template, carefully mark the center of all module-mounting holes on the panel.
3. Return the assembled modules to the clean work surface, including any previously mounted modules.
4. Drill and tap the mounting holes for the recommended M4 or #8 screw.
5. Place the modules back on the panel, and check for proper hole alignment.
6. Attach the modules to the panel using the mounting screws.

NOTE	If mounting more modules, mount only the last one of this group and put the others aside. This reduces remounting time during drilling and tapping of the next group.

7. Repeat steps 1 to 6 for any remaining modules.

DIN Rail Mounting

The module can be mounted using the following DIN rails:

- 35 × 7.5 mm (EN 50 022 – 35 × 7.5)
- 35 × 15 mm (EN 50 022 - 35 × 15)

Before mounting the module on a DIN rail, close the DIN rail latches. Press the DIN rail mounting area of the module against the DIN rail. The latches will momentarily open and lock into place.

2.6.1 Replacing a Single Module within a System

1. Remove power. See important note at the beginning of this chapter.
2. On the module to be removed, remove the upper and lower mounting screws from the module (or open the DIN latches using a flat-blade or Phillips head screwdriver).
3. Move the bus lever to the right to disconnect (unlock) the bus.
4. On the right-side adjacent module, move its bus lever to the right (unlock) to disconnect it from the module to be removed.
5. Gently slide the disconnected module forward. If you feel excessive resistance, check that the module has been disconnected from the bus, and that both mounting screws have been removed (or DIN latches opened).

NOTE	It may be necessary to rock the module slightly from front to back to remove it, or, in a panel-mounted system, to loosen the screws of adjacent modules.

6. Before installing the replacement module, be sure that the bus lever on the module to be installed and on the right-side adjacent module or end cap are in the unlocked (fully right) position.
7. Slide the replacement module into the open slot.
8. Connect the modules together by locking (fully left) the bus levers on the replacement module and the right-side adjacent module.
9. Replace the mounting screws (or snap the module onto the DIN rail).

Section 2.7 Wiring the Module

When wiring your system, use the following guidelines:

General

- Channels are isolated from one another by ±500 VDC maximum.
- As a general rule, allow at least 15.2 cm (6 in.) of separation for every 120 V of power.
- Routing field wiring in a grounded conduit can reduce electrical noise.
- If field wiring must cross AC or power cables, ensure that they cross at right angles.
- Provision shall be made to prevent the rated voltage being exceeded by the transient disturbances of more than 140%.

  The equipment shall be installed in an enclosure that provides a degree of protection not less than IP 54 in accordance with EN 60079-0 and used in an environment of not more than pollution degree 2. The enclosure shall be accessible only with the use of a tool.
- Grounding to earth is accomplished through mounting of modules on rail.
- Subject devices are for operation in Ambient Temperature Range: 0 °C to +60 °C.

Terminal Block

- For voltage and current sensors, use Belden 8761 shielded, twisted-pair wire (or equivalent) to ensure proper operation and high immunity to electrical noise.
- To ensure optimum accuracy, limit overall cable impedance by keeping a cable as short as possible. Locate the module as close to output devices as the application permits.

Grounding

- This product is intended to be mounted to a well-grounded mounting surface such as a metal panel. Additional grounding connections from the module's mounting tabs or DIN rail (if used) are not required unless the mounting surface cannot be grounded.
- Keep cable shield connections to ground as short as possible.
- Ground the shield drain wire at one end only. The typical location is as follows.

- If it is necessary to connect the shield drain wire at the module end, connect it to earth ground using a panel or DIN rail mounting screw.
- Refer to Industrial Automation Wiring and Grounding Guidelines, Allen-Bradley publication 1770-4.1, for additional information.

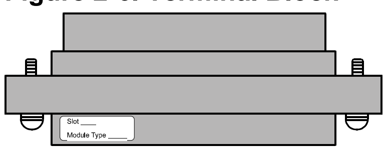
Terminal Door Label

A removable, write-on label is provided with the module. Remove the label from the door, mark your unique identification of each terminal with permanent ink, and slide the label back into the door. Your markings (ID tag) will be visible when the module door is closed.

Removing and Replacing the Terminal Block

When wiring the module, you do not have to remove the terminal block. If you remove the terminal block, use the write-on label located on the side of the terminal block to identify the module location and type.

Figure 2-6. Terminal Block



To remove the terminal block, loosen the upper and lower retaining screws. The terminal block will back away from the module as you remove the screws. When replacing the terminal block, torque the retaining screws to 0.46 Nm (4.1 in-lbs).

Wiring the Finger-Safe Terminal Block

1. Loosen the terminal screws to be wired.
2. Route the wire under the terminal pressure plate. You can use the bare wire or a spade lug. The terminals accept a 6.35 mm (0.25 in.) spade lug.

NOTE	The terminal screws are non-captive. Therefore, it is possible to use a ring lug [maximum ¼-inch o.d. with a 0.139 inch minimum i.d. (M3.5)] with the module.

3. Tighten the terminal screw making sure the pressure plate secures the wire. Recommended torque when tightening terminal screws is 0.68 Nm (6 in-lbs).

NOTE	If you need to remove the finger-safe cover, insert a screwdriver into one of the square wiring holes and gently pry the cover off. If you wire the terminal block with the finger-safe cover removed, you may not be able to put it back on the terminal block because the wires will be in the way.

Wire Size and Terminal Screw Torque

Each terminal accepts up to two wires with the following restrictions:

Wire Type	Wire Size	Terminal Screw Torque	Retaining Screw Torque
Solid Cu-90 °C (194 °F)	#14 to #22 AWG (1.63 to 0.65 mm)	0.68 Nm (6 in-lbs)	0.46 Nm (4.1 in-lbs)
Stranded Cu-90 °C (194 °F)	#16 to #22 AW (1.29 to 0.65 mm)	0.68 Nm (6 in-lbs)	0.46 Nm (4.1 in-lbs)

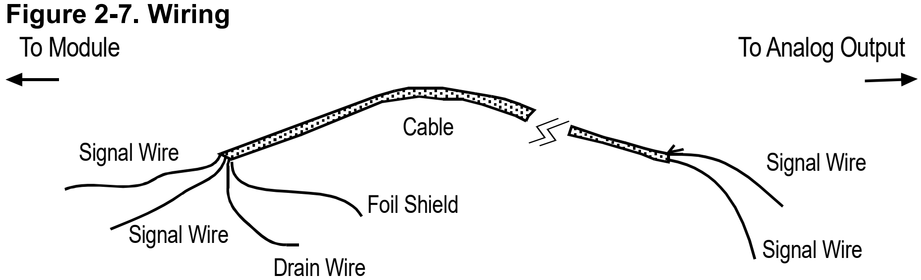
WARNING	USE SUPPLY WIRES SUITABLE FOR 20 °C ABOVE SURROUNDING AMBIENT TEMPERATURE.

WARNING	UTILISER DES FILS D'ALIMENTATION QUI CONVIENNENT A UNE TEMPERATURE DE 20 °C AU-DESSUS DE LA TEMPERATURE AMBIANTE.

WARNING	SHOCK HAZARD To prevent shock hazard, care should be taken when wiring the module to analog signal sources. Before wiring any module, disconnect power from the system power supply, and any other power source to the module.

After the module is properly installed, follow the wiring procedure below, using the proper cable, Belden 8761.

Figure 2-7. Wiring



To wire your module follow these steps.

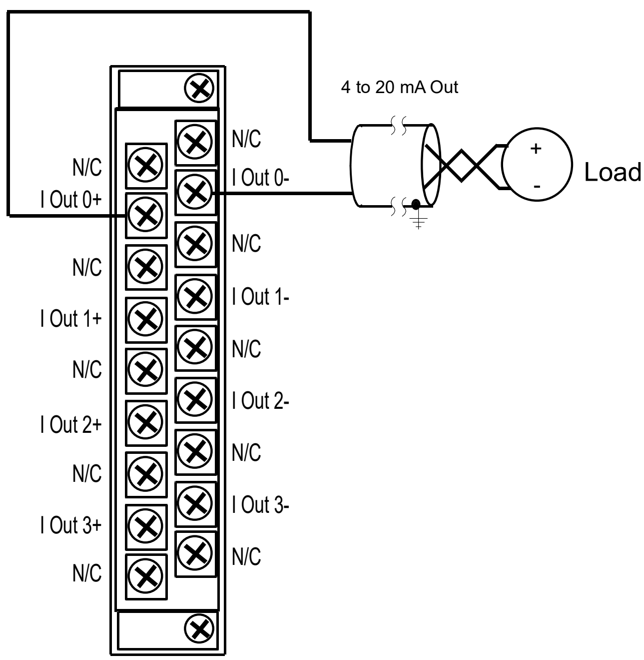
1. At each end of the cable, strip some casing to expose the individual wires.

WARNING	HAZARD OF DAMAGE TO EQUIPMENT Be careful when stripping wires. Wire fragments that fall into a module could cause damage at power up.

2. Trim the signal wires to 5 cm (2-inch) lengths. Strip about 5 mm (3/16-inch) of insulation away to expose the end of the wire.
3. At one end of the cable, twist the drain wire and foil shield together, bend them away from the cable, and apply shrink wrap. Then earth ground at the preferred location based on the type of sensor you are using. See Grounding for more details.
4. At the other end of the cable, cut the drain wire and foil shield back to the cable and apply shrink wrap.
5. Connect the signal wires to the terminal block. Connect the other end of the cable to the analog output device.
6. Repeat steps 1 through 5 for each channel on the module.

Wiring Diagram

Figure 2-8. Wiring Diagram



Calibration

The isolated HART module is initially calibrated at the factory.

2.7.1 Perform the Startup Procedure

1. Apply power to the controller system.
2. Download your program, which contains the module configuration settings, to the controller.
3. Put the controller in Run mode. During a normal start-up, the module status LED turns on.

NOTE	If the module status LED does not turn on, cycle power. If the condition persists, contact your local distributor or Spectrum Controls for assistance.

2.7.2 Monitor Module Status to Check if the Module is Operating Correctly

Module and channel configuration errors are reported to the controller. These errors are typically reported in the controller's I/O status file. Channel status data is also reported in the module's input data table, so these bits can be used in your control program to flag a channel error.

Chapter 3
Configuring the OF4IH for CompactLogix Using RSLogix 5000

This chapter explains how to incorporate the OF4IH module into a CompactLogix system using RSLogix 5000 programming software. The process of incorporating your HART module into the CompactLogix system is similar to the process needed to add an Allen-Bradley module. You use your RSLogix 5000 programming software to install and configure your HART module.

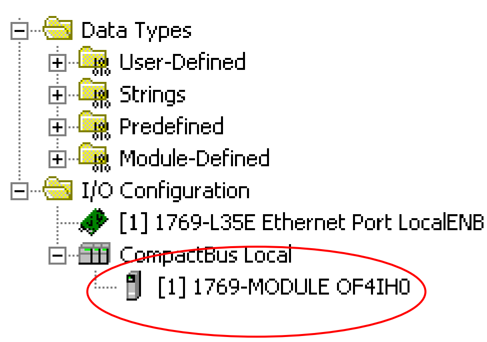
If you choose not to use the generic module profile, an Add-On profile is available on our website to help with the installation of the module. The Add-On profile download also includes an RSLogix 5000 sample project demonstrating how to read and write HART data to and from each channel. The sample project contains user defined data types, configuration tags, input tags, output tags, and ladder samples needed to configure each HART module. The topics discussed in this chapter include:

- Setting up the generic profile.
- Using the Add-On profile.
- Understanding user defined data types.
- Adding the controller and program tags.
- Using the provided ladder sample.

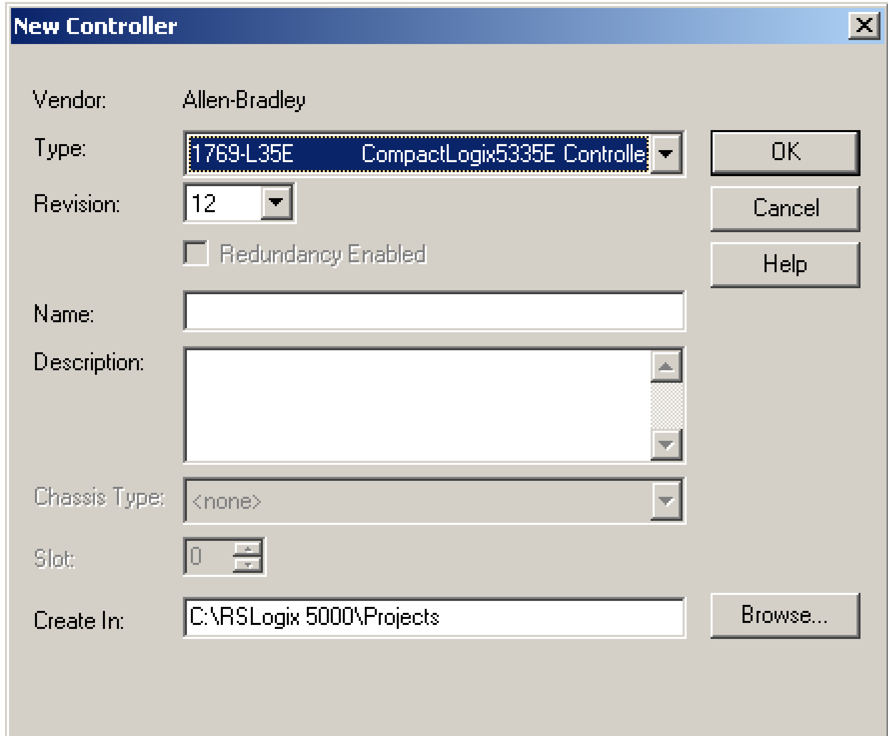
Section 3.1 Setting up the Generic Profile

The generic profile defines the module for the CompactBus, so that the right number of input, output, and configuration words are reserved. To configure the generic profile, you can use the profile already created in the sample project, see Figure 4-1, or follow the procedures outlined below.

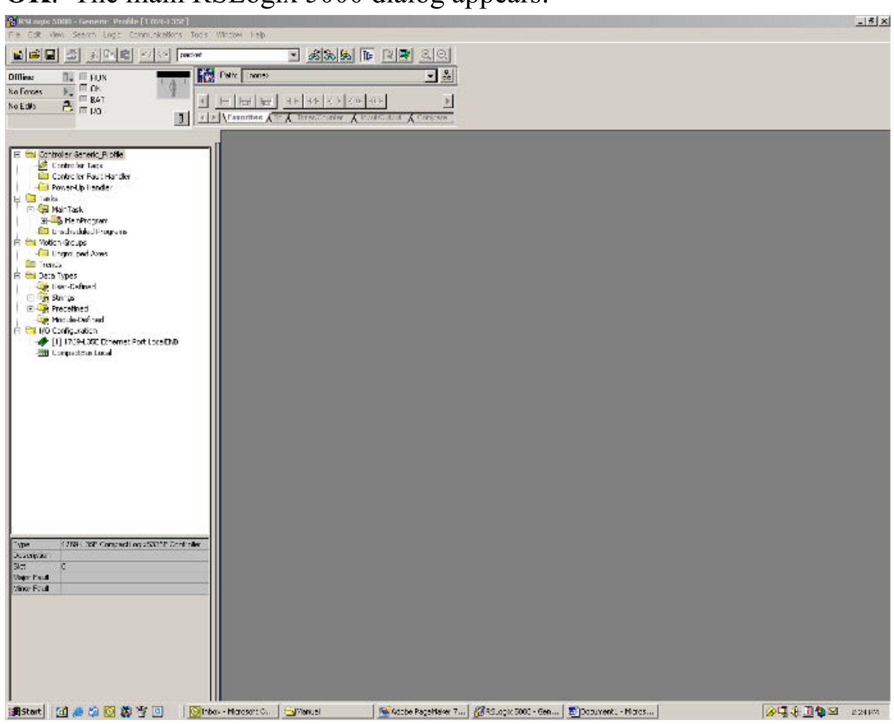
Figure 3-1. Module Information



1. Click on the new project icon or on the **File** pull-down menu and select **New**. The following dialog appears:

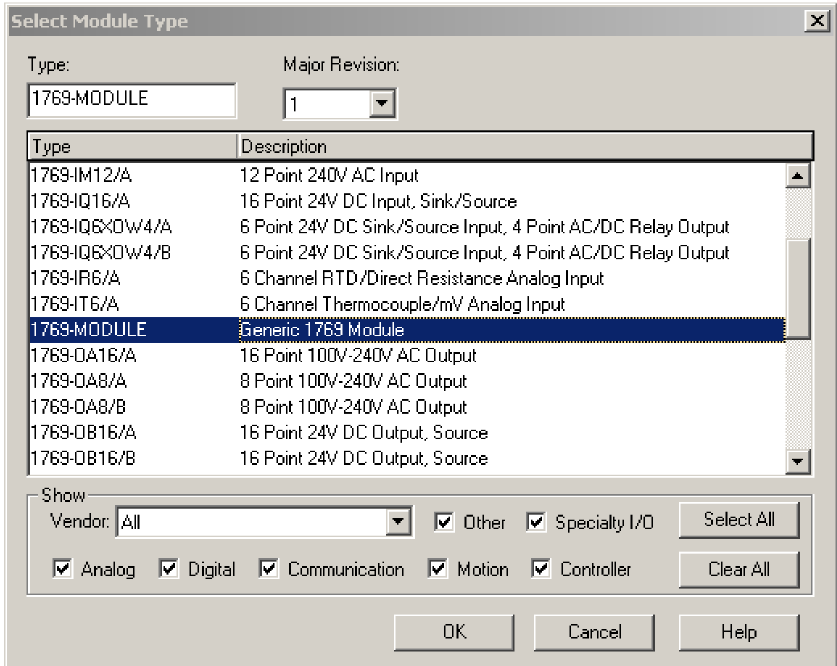


2. Choose your controller type, enter a name for your project, and click **OK**. The main RSLogix 5000 dialog appears:



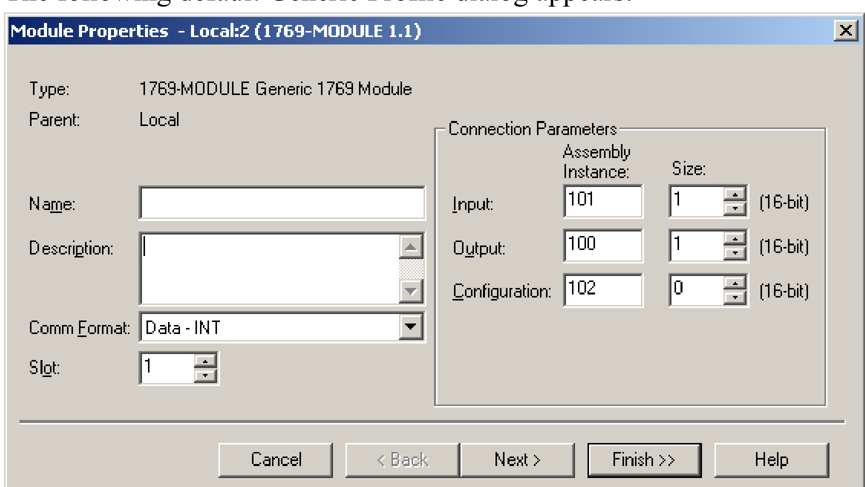
3. In the Controller Organizer on the left of the dialog, right click **[0]CompactBus Local**, and select **New Module**.

   The following dialog appears:



4. This dialog is used to narrow your search for I/O modules to configure in your system. With the initial release of the CompactLogix 5320 controller, this dialog only includes the **Generic 1769 Module**. Click the **OK** button.

   The following default Generic Profile dialog appears:



5. Select the Comm Format (**Data – INT** for the 1769sc-OF4IH). Enter a name in the **Name** field. In this example, **OF4IH** is used to help identify the module type in the Controller Organizer. The **Description** field is optional and may be used to provide more details concerning this I/O module in your application.

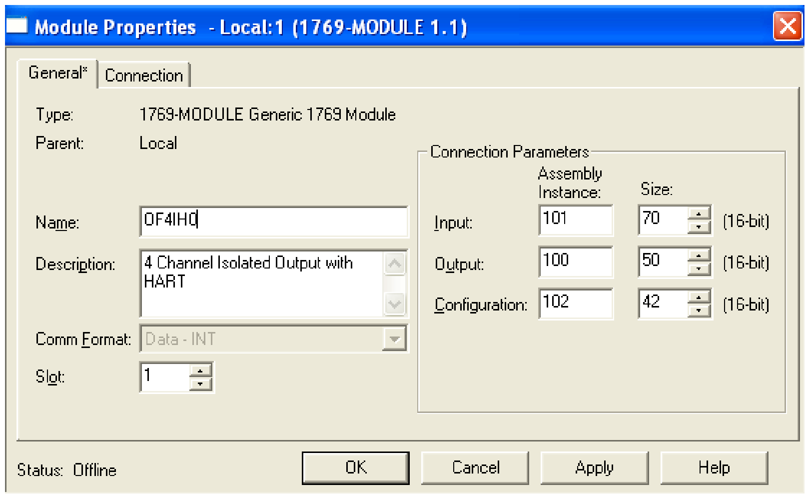
6. Next, select the slot number, although it will begin with the first available slot number, 1, and increments automatically for each subsequent Generic Profile you configure. For this example, the 1769sc-OF4IH HART module is in slot 1.

   The **Comm Format**, **Assembly Instance**, and **Size** values are listed in the following table for the 1769sc-OF4IH HART module:

Table 3-1. Generic Profile Parameters

1769 I/O Module	Comm Format	Parameter	Assembly Instance	Size (16-Bit)
OF4IH	Data-INT	Input Output Config	101 100 102	70 50 42

7. Enter the **Assembly Instance** numbers and their associated sizes for the 1769sc-OF4IH module into the Generic Profile. When complete, the Generic Profile for a 1769sc-OF4IH module should look like the following:



8. Click **Finish** to complete the configuration of your I/O module.
9. Configure each I/O module in this manner. The CompactLogix 5320 controller supports a maximum of 8 I/O modules. The valid slot numbers to select when configuring I/O modules are **1** through **8**.

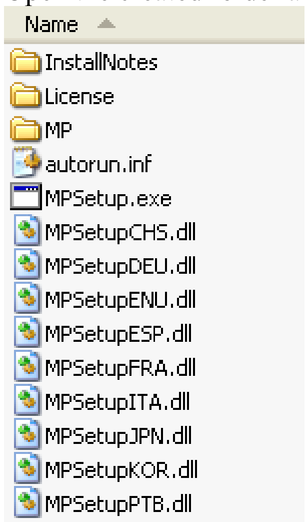
Section 3.2 Using The Add-On Profile

For RSLogix 5000 version 15 and greater, an Add-On module profile is available for download at www.spectrumcontrols.com. The Add-On profile allows the user to add the OF4IH module to the RSLogix 5000 module pick list. The profile provides configuration and information dialogs to the user, to simplify installation. Follow the procedure below to install and use the Add-On profile.

NOTE	Module firmware 2.0 and greater is required to use the Add-On profile.

3.2.1 Installing the Add-On profile

1. Download the zipped file, sc1769_hart_15.zip, from the Spectrum Controls website and unzip the file.
2. Open the created folder and double-click on the MPSetup.exe file.

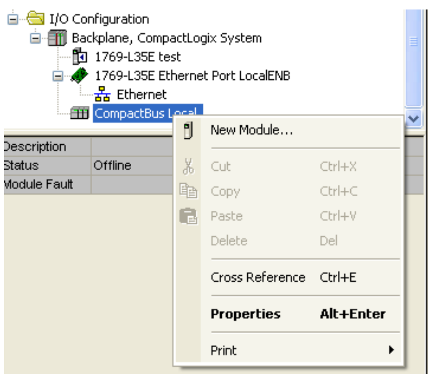


3. Follow the online prompts.

3.2.2 Adding the OF4IH Module to Your Logix Project

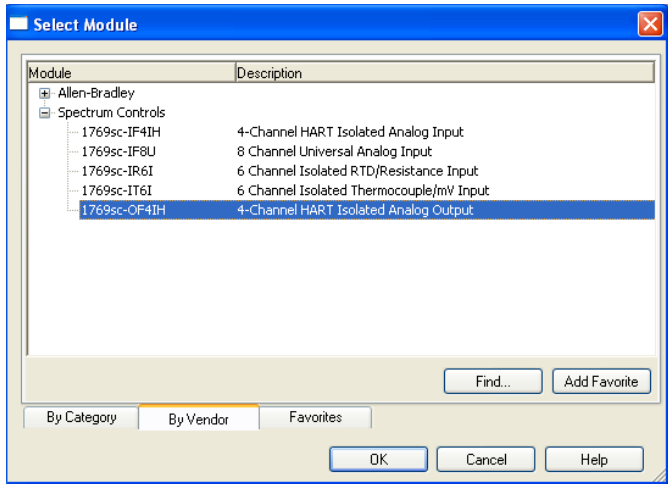
Once the profiles are installed you can access them through RSLogix 5000 via the I/O Configuration. Follow the procedure below to add a module:

1. In the I/O Configuration, right click on the 1769 CompactBus and select **New Module**:



2. When the dialog opens, select the **By Vender** tab, and expand the Spectrum Controls folder.
3. Highlight the module and click **OK**.

4. Configure the module using the custom configuration dialogs:

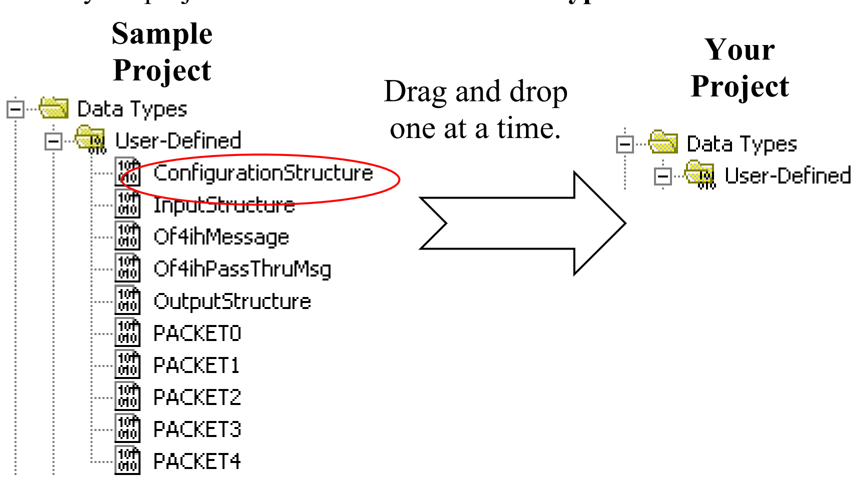


NOTE	The 1769sc-OF4IH still requires ladder to demultiplex the HART data, and to send HART messages via the controller. Please refer to the sample project packaged with the profile install for more information.

Section 3.3 User Defined Data Types

The sample project contains user defined data types that define the structure for tags used within the project. The data types organize the HART data returned by the module and are referenced throughout this manual, so it is highly recommended that these data types be used whenever possible.

Select the data type you wish to copy from the **Controller Organizer** and paste it into your project under **User-Defined Data Types**:



NOTE	The user-defined data types should be copied before copying the tags or ladder.

The table below gives a brief description of each data type.

Table 3-2. User Defined Data Type Descriptions

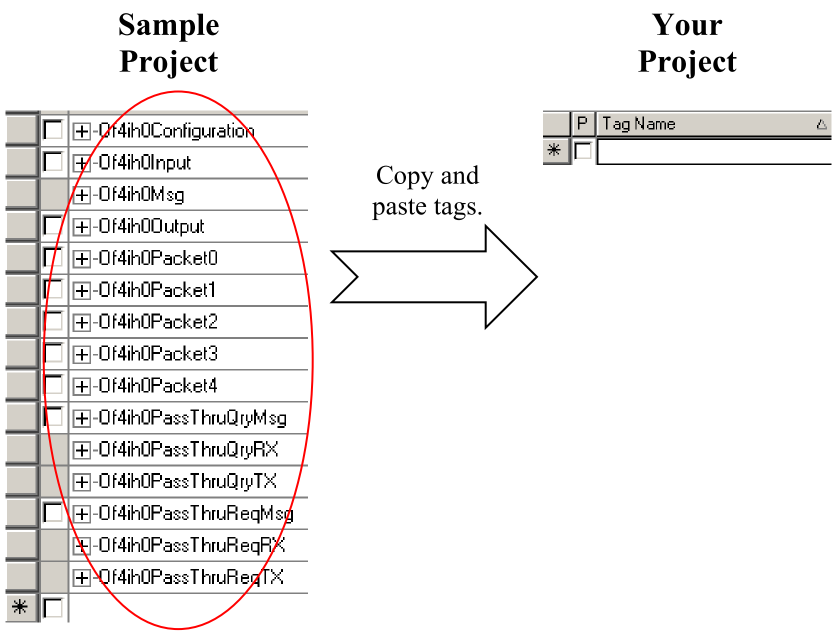
User-Defined Data Type	Description
ConfigurationStructure	Defines the structure for the configuration tags used to configure the module.[1]
GetDeviceInfoStructure	Defines the structure of the HART data returned by the module when the module specific command, *Get Device Information*, is sent to module.[2]
OF4IHMessage	This data type defines the structure for tags used to send messages to and from the module using the paging scheme.[2]
OF4IHPassThruMsg	Defines the structure for tags used to send HART pass through messages to and from the module.[2]
InputStructure	Defines the structure for the input tags returned by the module.[1]
OutputStructure	Defines the structure for the output tags used by the module.[1]
Packet0	Defines the data structure for HART packet 0. HART packet zero contains device information for the connected HART device.[2]
Packet1	Defines the data structure for HART packet 1. HART packet 1 is used to display the four dynamic variables for the selected HART device.[2]
Packet2	Defines the data structure for HART packet 2. HART packet 2 is used to display the slot variables for the connected HART device.[2]
Packet3	Defines the data structure for HART packet 3. HART packet 3 displays the ASCII message for the connected HART device.[2]
Packet4	Defines the data structure for HART packet 4. HART packet 4 contains the extended status for the connected HART device.[2]

Section 3.4 Project Tags

The project tags were created to simplify the configuration of the module as well as reduce confusion related to using only the module local tags. The tags defined in the sample project utilize the *user defined data types* described in the previous section.

The tags from the controller scope should be copied to your project before the tags contained in the individual program sections. Open the controller tags on the sample project and select the *edit* tags mode. Grab the tags you want to copy by using the left mouse button and dragging. See figure below.

[1] Refer to Chapter 6 for more details.
[2] Refer to Chapter 7 for more details.



After copying the controller tags you can copy the program tags next. Follow the same procedure shown in Figure 4-8.

The figures below show examples of the configuration tags, input tags and output tags. Refer to Chapter 5 for information on how to configure the module and or reading the input data.

Refer to Chapter 6 for information regarding HART packet tags and pass-through tags:

⊟-Of4ih0Configuration	{...}	{...}		ConfigurationStructure
⊞-Of4ih0Configuration.Ch0ConfigWord0	2#1000_0000_0000_0000		Binary	INT
⊞-Of4ih0Configuration.Ch0ConfigWord1	2#0000_0001_0000_0001		Binary	INT
⊞-Of4ih0Configuration.Ch0FaultValue	4000		Decimal	INT
⊞-Of4ih0Configuration.Ch0ProgramIdleValue	4000		Decimal	INT
⊞-Of4ih0Configuration.Ch0LowClamp	0		Decimal	INT
⊞-Of4ih0Configuration.Ch0HighClamp	0		Decimal	INT
⊞-Of4ih0Configuration.Ch0RampRate	0		Decimal	INT
⊞-Of4ih0Configuration.Ch0Reserved	0		Decimal	INT
⊞-Of4ih0Configuration.Ch1ConfigWord0	2#0000_0000_0000_0000		Binary	INT
⊞-Of4ih0Configuration.Ch1ConfigWord1	2#0000_0000_0000_0000		Binary	INT
⊞-Of4ih0Configuration.Ch1FaultValue	0		Decimal	INT
⊞-Of4ih0Configuration.Ch1ProgramIdleValue	0		Decimal	INT
⊞-Of4ih0Configuration.Ch1LowClamp	0		Decimal	INT
⊞-Of4ih0Configuration.Ch1HighClamp	0		Decimal	INT
⊞-Of4ih0Configuration.Ch1RampRate	0		Decimal	INT
⊞-Of4ih0Configuration.Ch1Reserved	0		Decimal	INT
⊞-Of4ih0Configuration.Ch2ConfigWord0	2#0000_0000_0000_0000		Binary	INT
⊞-Of4ih0Configuration.Ch2ConfigWord1	2#0000_0000_0000_0000		Binary	INT
⊞-Of4ih0Configuration.Ch2FaultValue	0		Decimal	INT
⊞-Of4ih0Configuration.Ch2ProgramIdleValue	0		Decimal	INT
⊞-Of4ih0Configuration.Ch2LowClamp	0		Decimal	INT
⊞-Of4ih0Configuration.Ch2HighClamp	0		Decimal	INT
⊞-Of4ih0Configuration.Ch2RampRate	0		Decimal	INT
⊞-Of4ih0Configuration.Ch2Reserved	0		Decimal	INT
⊞-Of4ih0Configuration.Ch3ConfigWord0	2#0000_0000_0000_0000		Binary	INT
⊞-Of4ih0Configuration.Ch3ConfigWord1	2#0000_0000_0000_0000		Binary	INT
⊞-Of4ih0Configuration.Ch3FaultValue	0		Decimal	INT
⊞-Of4ih0Configuration.Ch3ProgramIdleValue	0		Decimal	INT
⊞-Of4ih0Configuration.Ch3LowClamp	0		Decimal	INT
⊞-Of4ih0Configuration.Ch3HighClamp	0		Decimal	INT
⊞-Of4ih0Configuration.Ch3RampRate	0		Decimal	INT
⊞-Of4ih0Configuration.Ch3Reserved	0		Decimal	INT
⊞-Of4ih0Configuration.Pad	0		Decimal	INT
⊞-Of4ih0Configuration.HartConfig	2#0000_0001_1111_1111		Binary	INT
⊞-Of4ih0Configuration.Ch0SlotVariable01	0		Decimal	INT
⊞-Of4ih0Configuration.Ch0SlotVariable23	0		Decimal	INT
⊞-Of4ih0Configuration.Ch1SlotVariable01	0		Decimal	INT

Figure 3-2. Input Tags

⊟-Of4ih0Input	{...}	{...}		InputStructure
⊞-Of4ih0Input.StatusWord0	2#0000_0000_0000_0000		Binary	INT
⊞-Of4ih0Input.StatusWord1	2#0000_0000_0000_0000		Binary	INT
⊞-Of4ih0Input.Ch0Data	4000		Decimal	INT
⊞-Of4ih0Input.Ch1Data	-32767		Decimal	INT
⊞-Of4ih0Input.Ch2Data	-32767		Decimal	INT
⊞-Of4ih0Input.Ch3Data	-32767		Decimal	INT
⊞-Of4ih0Input.HartData	{...}	{...}	Decimal	SINT[40]
⊞-Of4ih0Input.MsgSlaveControl	16#0101		Hex	INT
⊞-Of4ih0Input.MsgResponseSize	16		Decimal	INT
⊞-Of4ih0Input.MsgResponseBuffer	{...}	{...}	Decimal	SINT[40]

Figure 3-3. Output Tags

⊟-Of4ih0Output	{...}	{...}		OutputStructure
⊞-Of4ih0Output.Ch0OutputValue	4000		Decimal	INT
⊞-Of4ih0Output.Ch1OutputValue	0		Decimal	INT
⊞-Of4ih0Output.Ch2OutputValue	0		Decimal	INT
⊞-Of4ih0Output.Ch3OutputValue	0		Decimal	INT
⊞-Of4ih0Output.HartSuspendAndAlarmUnlatch	0		Decimal	INT
⊞-Of4ih0Output.HartPacketJustScanned	0		Decimal	INT
⊞-Of4ih0Output.MsgMasterControl	16#0101		Hex	INT
⊞-Of4ih0Output.MsgRequestSize	3		Decimal	INT
⊞-Of4ih0Output.MsgRequestBuffer	{...}	{...}	Decimal	SINT[40]

Section 3.5 Sample Project Ladder

The ladder contained in the sample project is used to perform several different operations. The *main* routine in the M*ainProgram* is used to copy data from the user defined tags to the module local tags. This data includes input, output and configuration settings for the module.

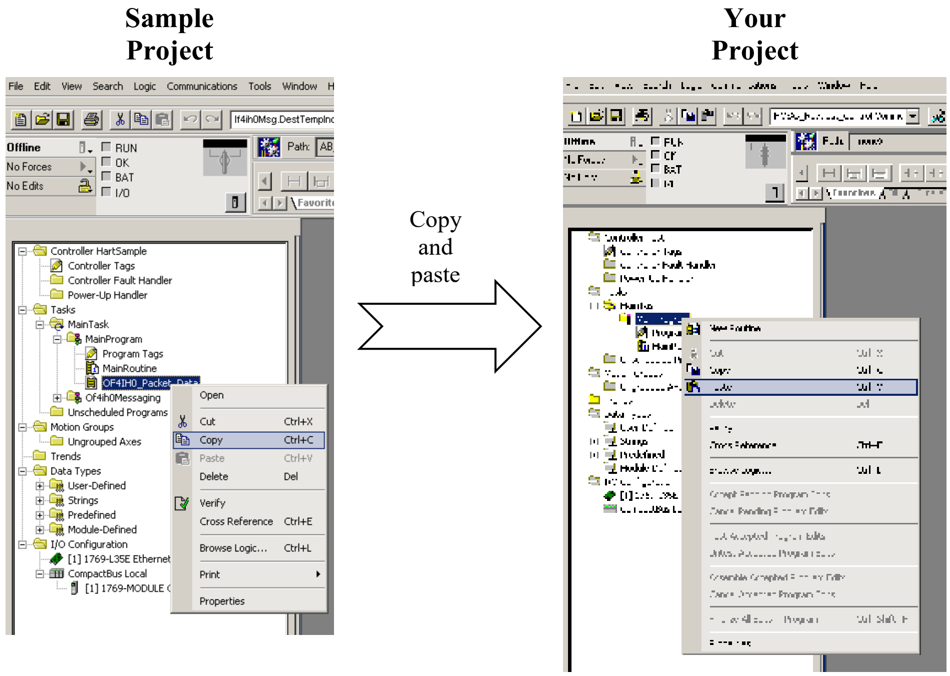
The *OF4IH0_Packet_Data* routine in the *MainProgram* contains the ladder that demultiplexes the HART data for each individual packet. Refer to Section 6.2 HART Packet Data for more information on HART and the HART packets.

The *OF4IH0Messaging* program contains several routines needed to send and receive HART messages to and from the module and the connected HART devices.

To copy any of the ladder, programs or routines, follow the procedure below:

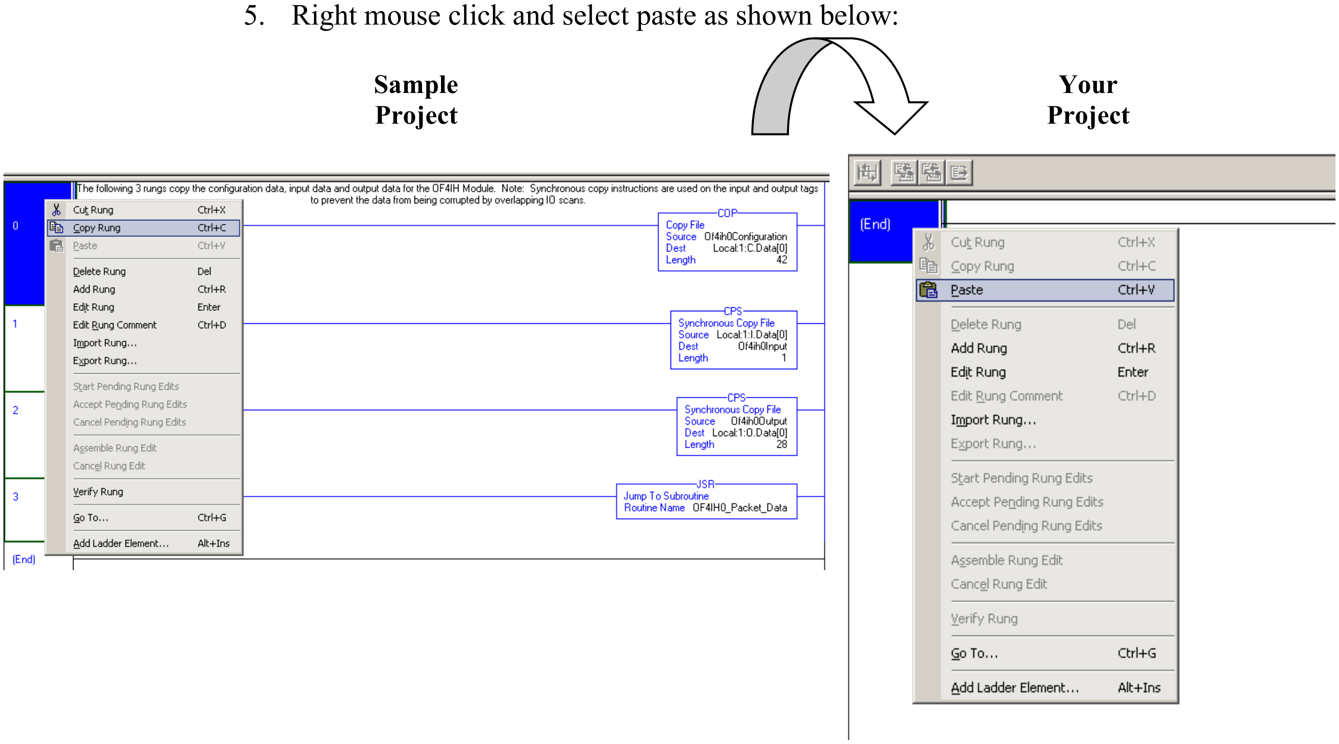
1. Select the program or routine.
2. Right mouse click and select copy.
3. Go to your project and select the appropriate program or task to place the new routine or program.

4. Right mouse click and select paste as shown below:



You can follow a similar procedure for copying ladder as well.

1. Open the routine that contains the ladder you want to copy.
2. Select the rungs to copy.
3. Right mouse click and select copy.
4. Open the routine in your project where you wish to paste the new rungs.
5. Right mouse click and select paste as shown below:



Chapter 4
Configuring the OF4IH for a MicroLogix 1500 Using RSLogix 500

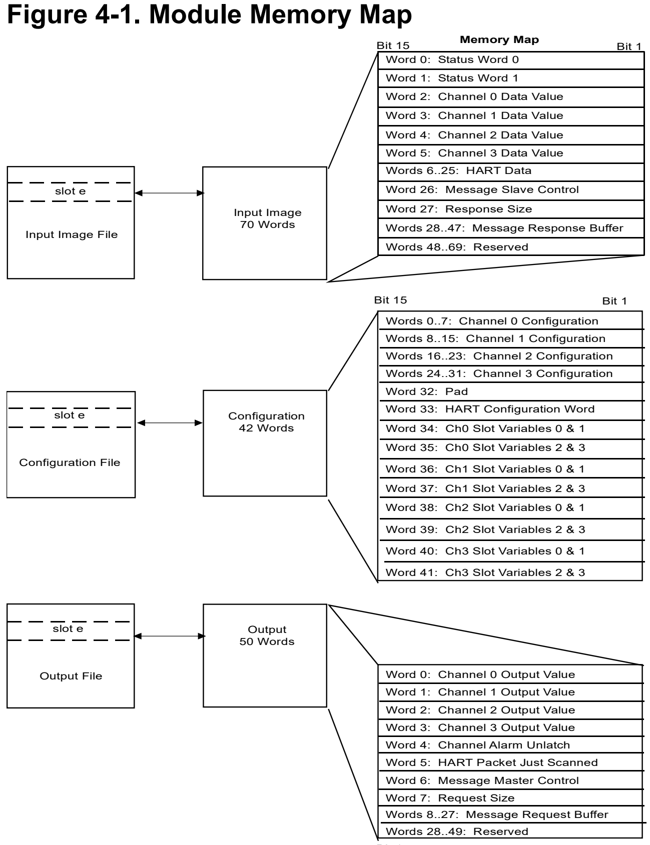
This chapter examines the 1769sc-OF4IH module's addressing scheme and describes module configuration using RSLogix 500 and a MicroLogix 1500 controller. This chapter will cover the following:

- Module Addressing
- Configuring the OF4IH in a MicroLogix 1500 System
- Using the Ladder Sample

Section 4.1 Module Addressing

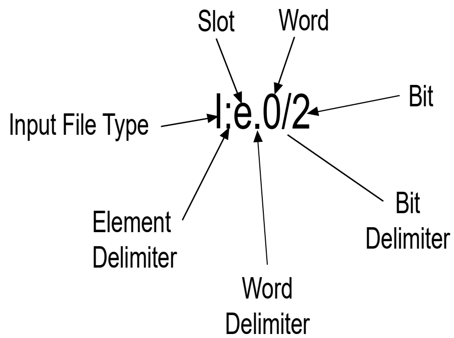
The following memory map shows the input, output, and configuration image tables for the module. Detailed information on the image table is in Chapter 5.

Figure 4-1. Module Memory Map



For example, to obtain the general status for channel 2 of the module located in slot e, use address I:e.0/2.

Figure 4-2. Address Example



NOTE	The end cap does not use a slot address.

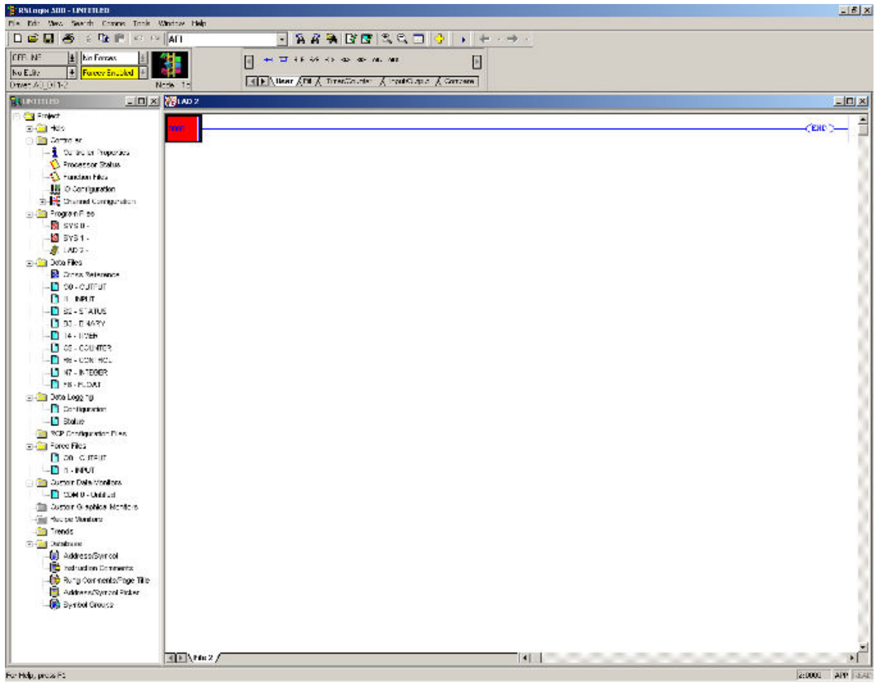
Section 4.2 Configuring the 1769sc-OF4IH in a MicroLogix 1500 System

This example takes you through configuring your 1769sc-OF4IH isolated HART analog output module with RSLogix 500 programming software, assumes your module is installed as expansion I/O in a MicroLogix 1500 system, and that RSLinx™ is properly configured and a communications link has been established between the MicroLogix processor and RSLogix 500.

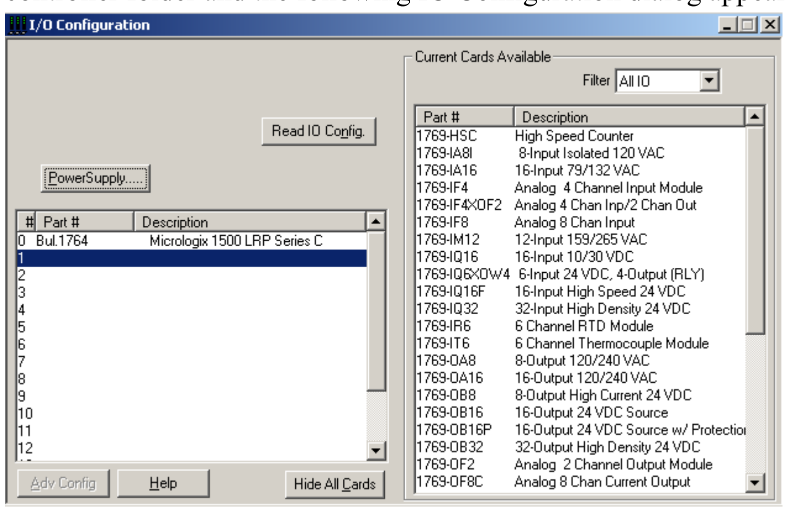
NOTE	It is recommended that a 1764-LRP series C processor with firmware version 5 or higher be used. The LRP processor supports floating point files, which is required to read floating point data from the IF4IH.

To configure:

1. Start RSLogix and create a MicroLogix 1500 application. The following dialog appears:

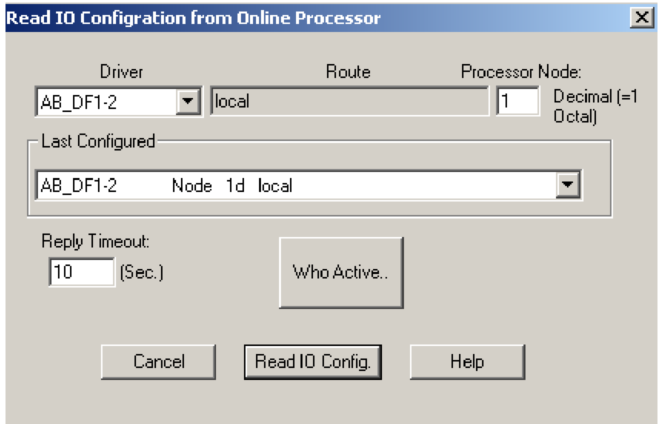


2. While offline, double-click on the **IO Configuration** icon under the controller folder and the following IO Configuration dialog appears.

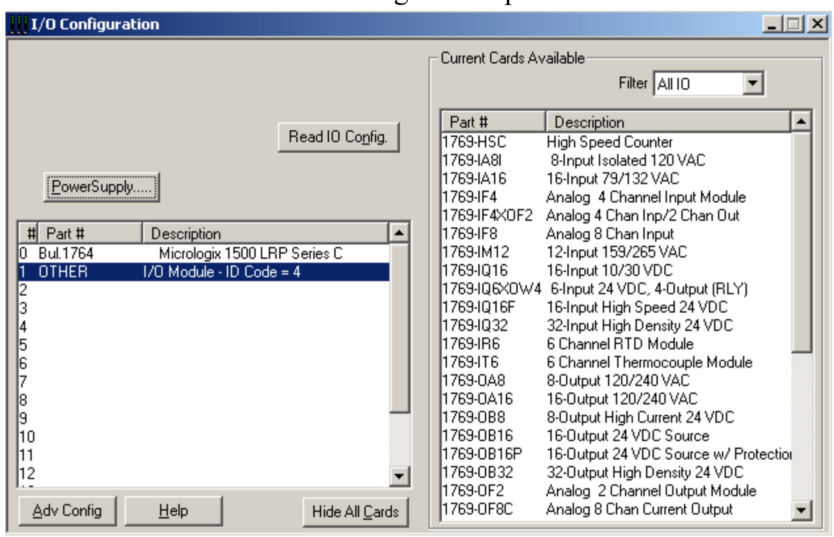


3. This dialog allows you to manually enter expansion modules into expansion slots, or to automatically read the configuration of the controller. To read the existing controller configuration, click on the **Read IO Config** button.

4. A communications dialog appears, identifying the current communications configuration so that you can verify the target controller. If the communication settings are correct, click on **Read IO Config**:

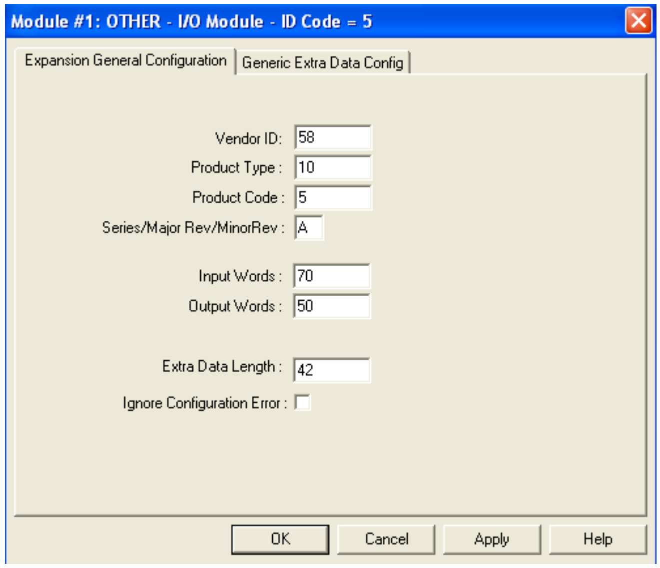


The actual I/O configuration is displayed. In this example, a second tier of I/O is attached to the MicroLogix 1500 processor.



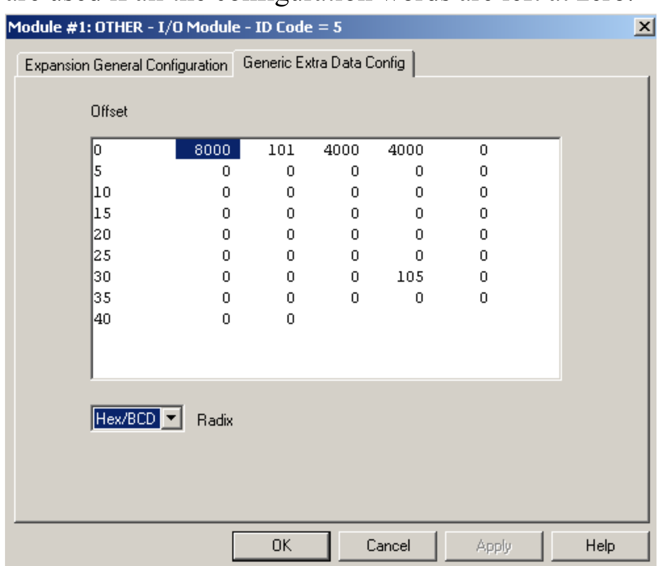
5. The 1769sc-OF4IH module is installed in slot 1. To configure the module, double-click on the module/slot.

The general configuration dialog appears.



NOTE	When using the read IO configuration feature in RSLogix, you need to manually enter 42 into the **Extra Data Length** field.

6. To configure the module, select the **Generic Extra Data Configuration** tab. Enter the decimal equivalent of each configuration word. There is a total of 42 words that need to be configured. The module default settings are used if all the configuration words are left at zero.



For a complete description of each of these parameters and the choices available for each of them, refer to Chapter 5.

Section 4.3 Using the Ladder Sample

To get started we recommend that you use the provided MicroLogix 1500 sample project. Refer to Chapter 7 for the sample project or visit our website at www.spectrumcontrols.com.

The sample project contains nine different subroutines which are used to perform various HART related tasks. The following list describes the function of each subroutine within the project file.

Table 4-1. Ladder Routines

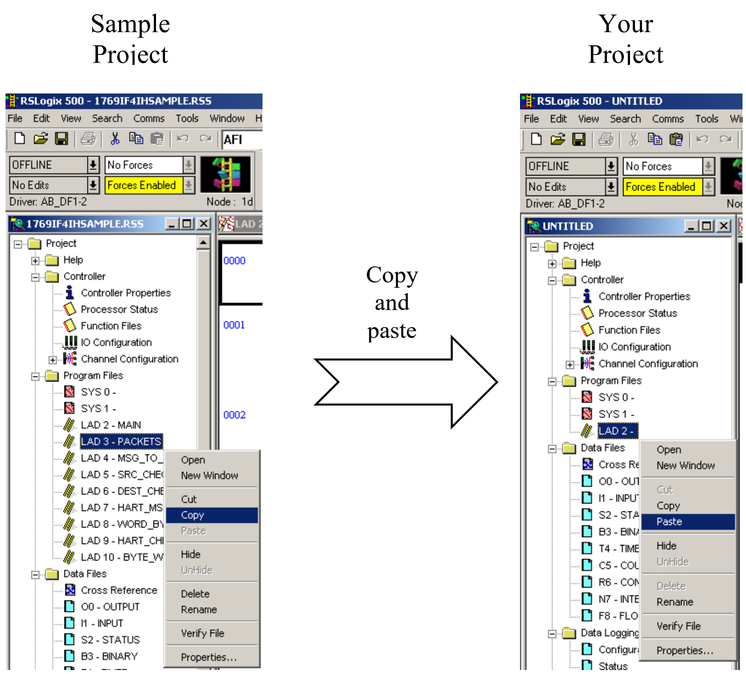
Routine	Description
MAIN	The main routine is the starting point for the ladder program.
PACKETS	The “packets” routine is used to demultiplex the HART data from the input file to individual integer files, so that the data can be viewed or used within the ladder program. This routine is called from the MAIN routine.
MSG_TO_MOD	This routine is used to send and receive messages to and from the module. Refer to Chapter 6 for more details regarding sending and receiving messages. This routine is called from the HART_MSG routine.
SRC_CHECK	Calculates the checksum for a message sent to the module one page at a time. This routine is called from the MSG_TO_MOD routine.
DEST_CHECKSUM	This routine calculates the checksum for a message received from the module one page at a time. This routine is called from the MSG_TO_MOD routine.
HART_MSG	This routine composes HART messages that will be sent to the module/field transmitter. This routine is called from the MAIN routine.
WORD_BYTE	Converts word data to its byte equivalent. This routine is called from the HART_MSG routine.
HART_CHECK	Calculates the checksum for the HART message being sent to the module/field device. This routine is called from the HART_MSG routine.
BYTE_WORD	Converts byte data to its word equivalent. This routine is called by the HART_MSG routine.

You have the choice to either use the sample project or copy and paste the pieces you need from the project.

4.3.1 Copying Subroutines from the Sample Project

To copy subroutines from the sample project to your project, follow the steps below:

1. Open the sample project and your project.
2. Select the subroutine you wish to copy.
3. Right mouse click and select copy.
4. Go to your project and select where you would like to place the new routine.
5. Right mouse click and select paste.

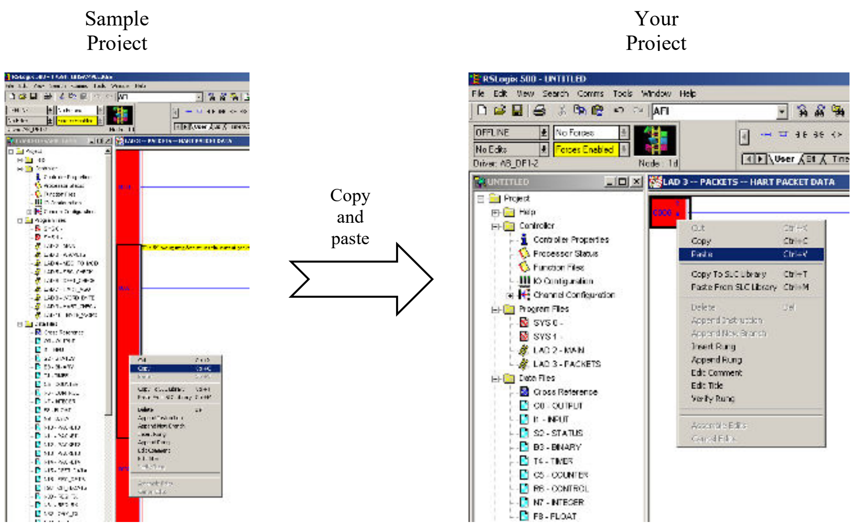


4.3.2 Copying Ladder from the Sample Project

To copy ladder, follow the procedure below:

1. Open the sample project and your project.
2. Open the routine that you wish to copy the ladder from.
3. Select the rungs by clicking the left mouse button. To select more rungs, select the first rung you wish to copy and while holding the shift key, select the last rung you wish to copy.
4. Right mouse click and select copy.
5. Open the routine in your project where you wish to place the new rungs.
6. Select the paste point by left mouse clicking.
7. Right mouse click and select paste.

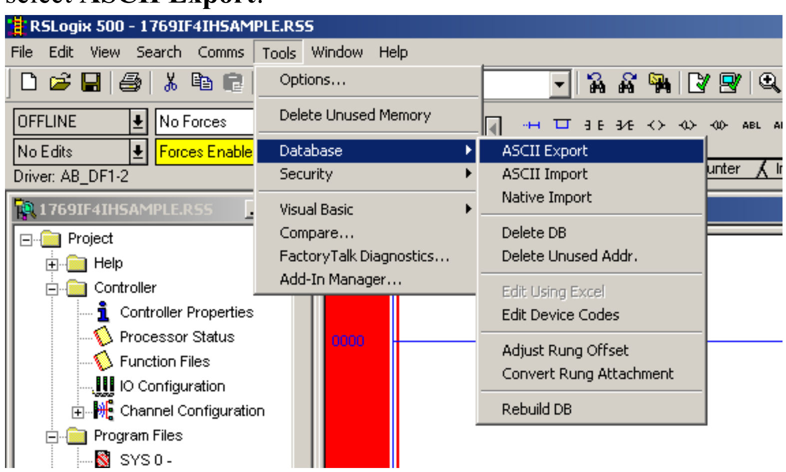
Figure 4-3. Copying Ladder



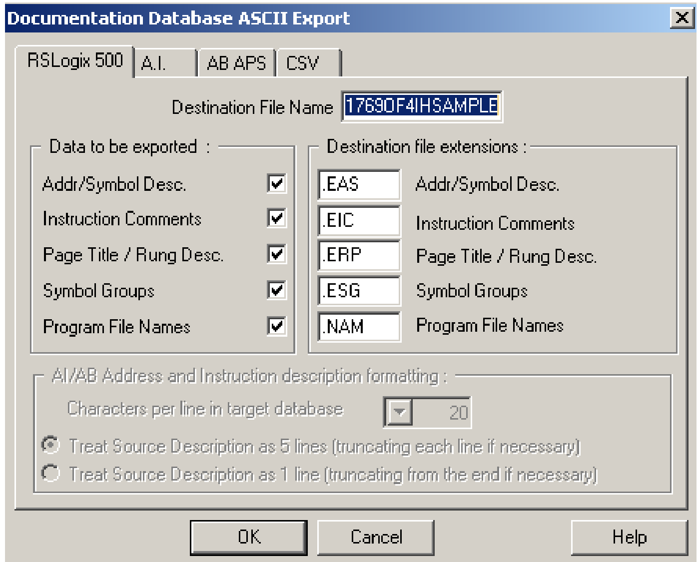
4.3.3 Importing Tag Database and Rung Comments

After copying the subroutines and or the ladder, you may wish to import the tags and rung comments. Follow the procedure below to import the tag database and rung comments:

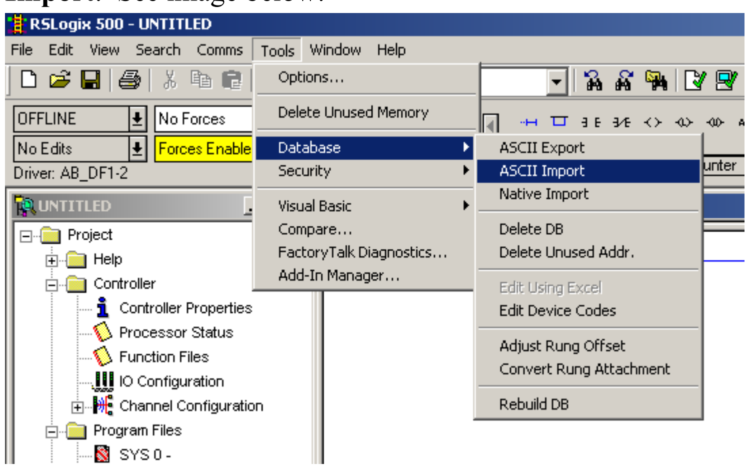
1. Open the sample project and your project.
2. In the sample project, from the Tools menu, select **Database**, and then select **ASCII Export**:



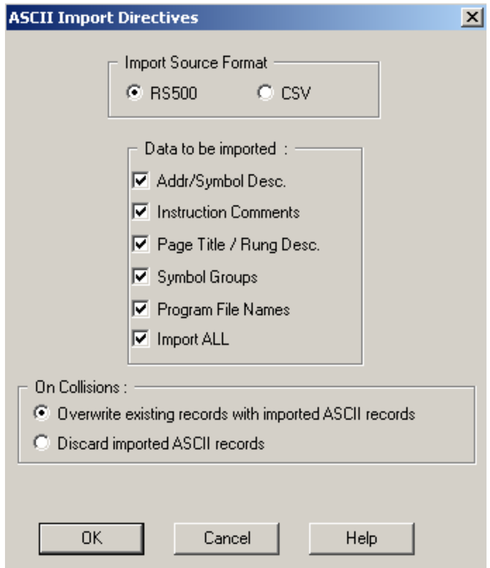
The following dialog appears:



3. Select the RSLogix 500 tab and click **OK**.
4. Select the location for the export file.
5. In your project, from the Tools menu, select **Database**, and select **ASCII Import**. See image below:



After selecting ASCII import the following dialog should appear:



6. Select the RSLogix 500 radio button and leave everything else at default. After making your selections, press the **OK** button.
7. Select the export file from steps 4 and 5 and click **Open**. You may be prompted for multiple files depending on the selections you made in step 7.

Chapter 5
Module Data, Status, and Channel Configuration

After installing the 1769sc-OF4IH isolated HART output module, you must configure it for operation, usually using the programming software compatible with the controller (for example, RSLogix 500 or RSLogix 5000). Once configuration is complete and reflected in the ladder logic, you need to operate the module and verify its configuration.

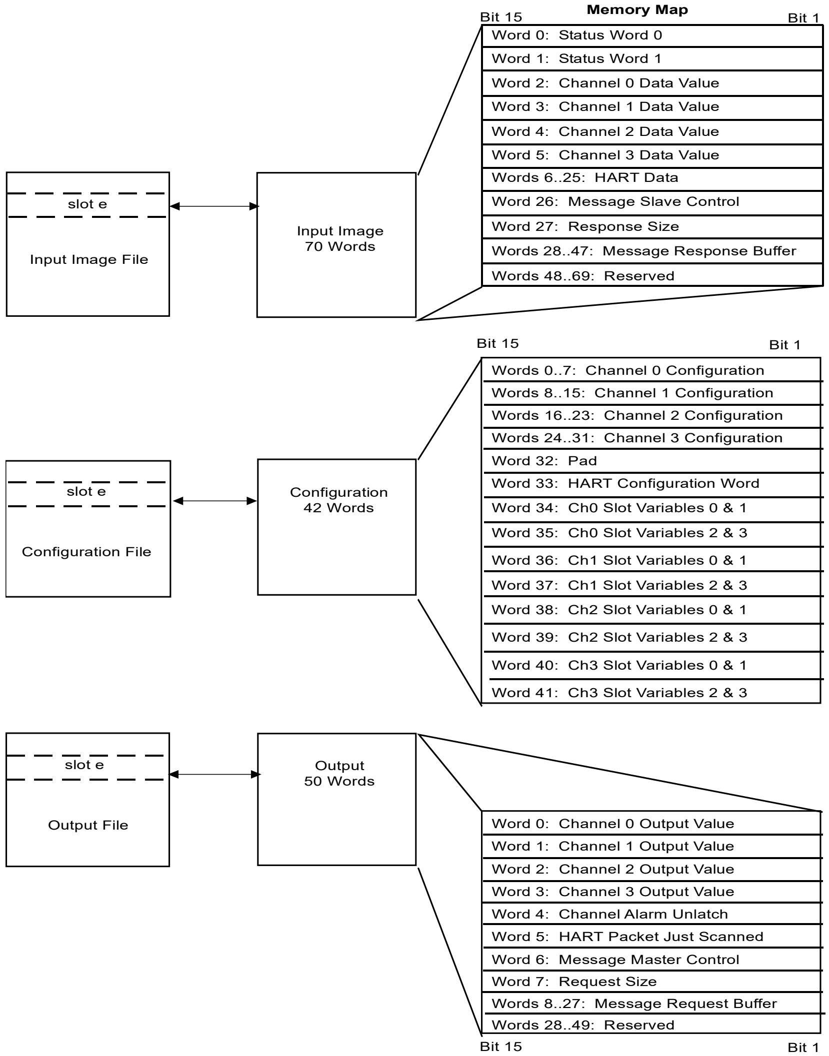
This chapter contains information on the following:

- Module memory map
- Accessing input image file data
- Configuring channels
- Determining effective resolution and range
- Determining module update time

Section 5.1 Module Memory Map

The module uses forty-eight input words for data and status bits (input image), twenty-eight output words, and forty-two configuration words.

Figure 5-1. Module Memory Map



NOTE	Not all controllers support program access to the configuration file. Refer to your controller's user manual.

Section 5.2 Accessing Input Image File Data

The input image file represents data words and status words. Words 0 and 1 contain status information for the four channels including process alarms and over and under range flags. Input words 2 through 5 hold an echo of the analog output value for channels 0 through 3 respectively. Words 6 through 25 include the HART packet data. Refer to Chapter 6 for information on how to demultiplex the HART packet data. Input word 26 holds the message control. Word 27 holds the message response size. Words 28 through 47 hold the message response buffer. Refer to Section 6.3 for more information regarding input words 28 through 47.

You can access the information in the input image file using the programming software configuration dialog. For information on configuring the module in a MicroLogix 1500 system using RSLogix 500, see Chapter 4; and for the CompactLogix using RSLogix 5000, see Chapter 3.

Section 5.3 Input Data File

The input data file allows you to access module input data for use in the control program, via word and bit access. The data table structure is shown in the table below.

Word/ Bit[1]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	OS3	OS2	OS1	OS0	Not Used								S3	S2	S1	S0
1	0	H3	U3	O3	0	H2	U2	O2	0	H1	U1	O1	0	H0	U0	O0
2	Channel 0 Command Value															
3	Channel 1 Command Value															
4	Channel 2 Command Value															
5	Channel 3 Command Value															
6..25	HART Packet Data															
26	Message Slave Control															
27	Message Response Size															
28..47	Message Response Buffer															
48..69	Reserved															

5.3.1 General Status Bits S0 to S3 (Word 0)

Bits S0 through S3 of word 0 contain the general status information for channels 0 through 3, respectively. If set (1), this bit indicates an error (over- or under-range, or output held).

5.3.2 Out of Service Status Bits OS0 to OS3 (Word 0)

Bits SO0 through SO3 of word 0 indicate whether the associated channel is *out of service* (that is, automatic HART acquisition is suspended).

NOTE	A channel that is placed out-of-service (that is, Suspended) will automatically resume service after three minutes, as long as no pass-through commands are issued before the three minutes expires.

Input Data Not Valid Condition

The general status bits S0 to S3 also indicate whether, or not, the input data for a

[1] Changing bit values is not supported by all controllers. Refer to your controller manual for details.

particular channel, 0 through 3, is being properly converted (valid) by the module. This “invalid data” condition can occur (bit set) when the download of a new configuration to a channel is accepted by the module (proper configuration) but before the A/D converter can provide valid (properly configured) data to the 1769 bus master/controller. The following information highlights the bit operation of the Data Not Valid condition.

1. The default and module power-up bit condition is reset (0).
2. The bit condition is set (1) when a new configuration is received and determined valid by the module. The set (1) bit condition remains until the module begins converting analog data for the previously accepted new configuration. When conversion begins, the bit condition is reset (0). The amount of time it takes for the module to begin the conversion process depends on the number of channels being configured and the amount of configuration data downloaded by the controller.

NOTE	If the new configuration is invalid, the bit function remains reset (0) and the module posts a configuration error. See Configuration Errors.

3. If A/D hardware errors prevent the conversion process from taking place, the bit condition is set (1).

5.3.3 Over-Range Flag Bits O0 to O3 (Word 1)

Over-range bits for channels 0 through 3 are contained in word 1. They apply to all output types. When set (1), the over-range flag bit indicates an output signal that is at the maximum of its normal operating range for the represented channel or sensor. The module automatically resets (0) the bit when the data value falls below the maximum for that range.

NOTE	This bit can also be set if the high clamp value is reached or exceeded. The bit will automatically clear when the channel command value[1] drops below the high clamp value.

5.3.4 Under-Range Flag Bits U0 to U3 (Word 1)

Under-range bits for channels 0 through 3 are contained in word 1. They apply to all output types. When set (1), the under-range flag bit indicates an output signal that is at the minimum of its normal operating range for the represented channel or sensor. The module automatically resets (0) when the under-range condition is cleared, and the data value is within the normal operating range.

[1] Module output words 0 through 3 for channel 0 through 3, respectively.

NOTE	This bit can also be set if the low clamp value is reached or exceeded. The bit will automatically clear when the channel command value[1] is above the low clamp value.

5.3.5 Hold Last State Bits H0 to H3 (Word 1)

The hold last state bit will be set on the associated channel if the channel is configured to output the last commanded value, and a fault or "run to program" condition is present.

5.3.6 Channel X[1] Command Value Echo (Words 2 to 5)

Data words 2 through 5 correspond to channels 0 through 3 respectively and contain the current command value[1] stored in the module's RAM memory for the associated channel. These input words can be used as a handshake to verify that the module has received the correct command value for the associated channel. The command value is defined later in this chapter.

5.3.7 HART Packet Data (Words 6 to 25)

This block of twenty words contains the multiplexed HART packet data for all four channels.[2]

5.3.8 Message Slave Control (Word 26)

The *message slave control* word controls how data is returned from the module after sending a message using output words 6 through 27[3].

5.3.9 Message Reply Size (Word 27)

The *message response size* indicates the number of bytes returned by the module after sending a message using output words 6 through 27**Error! Bookmark not defined.**.

5.3.10 Message Reply Buffer (Words 28…47)

After sending a message to the module, the response data for the message is stored in the *message response buffer*.

5.3.11 Reserved (Words 48…69)

Reserved for future expansion.

Section 5.4 Module Configuration

After module installation, you must configure operation details, such as output type, data format, etc., for each channel. Configuration data for the module is stored in the controller configuration file, which is both readable and writable. The default value of the configuration data is represented by zeros in the data

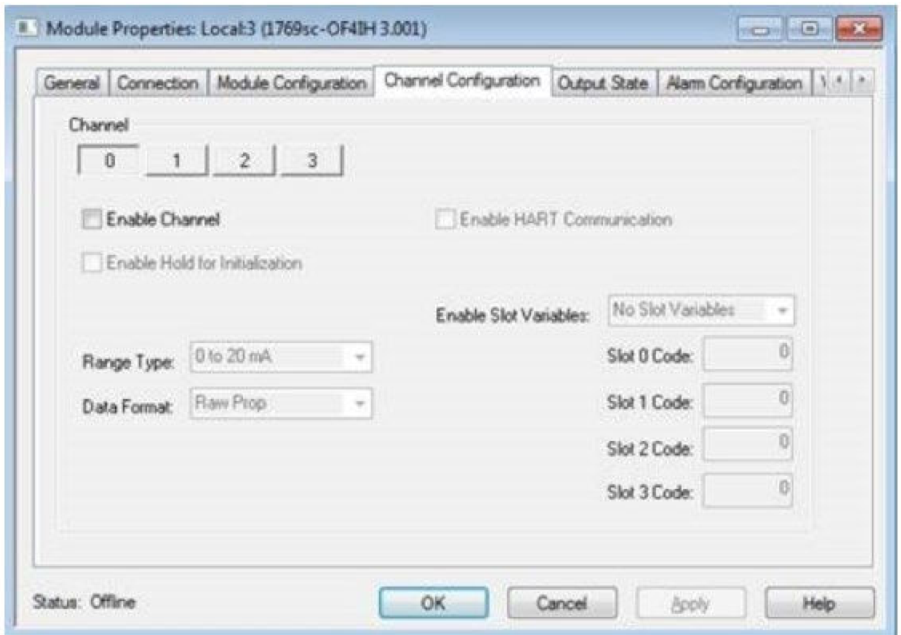
[1] X refers to channel number 0 through 3.
[2] For more details, refer to Chapter 6
[3] For more details, refer to Chapter 6

file.

Each channel defaults to:

- Channel disabled.
- Range type: 0 t0 20 mA.
- Data Format: Raw Proportional.
- HART Communication disabled.

See Associated dialog below:



The structure of the channel configuration file is shown below.

Table 5-1. Module Configuration

Word	Bit																Function
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	EC	Reserved			SV3	SV2	SV1	SV0	SIU	SIO	LA	ER	FM	PM	HI	PFE	Ch0 General Settings
1	Reserved					Data Format			Reserved					Output Type			Ch0 Output type and Data Format
2	Channel 0 Fault Value																Ch0 Fault Value
3	Channel 0 Program Idle Value																Ch0 Program Idle Value
4	Channel 0 Low Clamp Value																Ch0 Low Clamp Value
5	Channel 0 High Clamp Value																Ch0 High Clamp Value
6	Channel 0 Ramp Rate Value																Ch0 Ramp Rate Value
7	Reserved																Not Used (Set to Zero)
8	EC	Reserved			SV3	SV2	SV1	SV0	SIU	SIO	LA	ER	FM	PM	HI	PFE	Ch1 General Settings

Word	Bit																Function
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
9	Reserved					Data Format			Reserved					Output Type			Ch1 Output type and Data Format
10	Channel 1 Fault Value																Ch1 Fault Value
11	Channel 1 Program Idle Value																Ch1 Program Idle Value
12	Channel 1 Low Clamp Value																Ch1 Low Clamp Value
13	Channel 1 High Clamp Value																Ch1 High Clamp Value
14	Channel 1 Ramp Rate Value																Ch1 Ramp Rate Value
15	Reserved																Not Used (Set to Zero)
16	EC	Reserved			SV3	SV2	SV1	SV0	SIU	SIO	LA	ER	FM	PM	HI	PFE	Ch2 General Settings
17	Reserved					Data Format			Reserved					Output Type			Ch2 Output type and Data Format
18	Channel 2 Fault Value																Ch2 Fault Value
19	Channel 2 Program Idle Value																Ch2 Program Idle Value
20	Channel 2 Low Clamp Value																Ch2 Low Clamp Value
21	Channel 2 High Clamp Value																Ch2 High Clamp Value
22	Channel 2 Ramp Rate Value																Ch2 Ramp Rate Value
23	Reserved																Not Used (Set to Zero)
24	EC	Reserved			SV3	SV2	SV1	SV0	SIU	SIO	LA	ER	FM	PM	HI	PFE	Ch3 General Settings
25	Reserved					Data Format			Reserved					Output Type			Ch3 Output type and Data Format
26	Channel 3 Fault Value																Ch3 Fault Value
27	Channel 3 Program Idle Value																Ch3 Program Idle Value
28	Channel 3 Low Clamp Value																Ch3 Low Clamp Value
29	Channel 3 High Clamp Value																Ch3 High Clamp Value
30	Channel 3 Ramp Rate Value																Ch3 Ramp Rate Value

Word	Bit																Function
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
31	Reserved																Not Used (Set to Zero)
32	Set to Zero																Pad (16 Bit Alignment)
33	Reserved		PA		EH3	EH2	EH1	EH0	Handle Timeout								HART Configuration Word
34	Channel 0 HART Slot Variables 0 & 1																Defines Slot Variables
35	Channel 0 HART Slot Variables 2 & 3																Defines Slot Variables
36	Channel 1 HART Slot Variables 0 & 1																Defines Slot Variables
37	Channel 1 HART Slot Variables 2 & 3																Defines Slot Variables
38	Channel 2 HART Slot Variables 0 & 1																Defines Slot Variables
39	Channel 2 HART Slot Variables 2 & 3																Defines Slot Variables
40	Channel 3 HART Slot Variables 0 & 1																Defines Slot Variables
41	Channel 3 HART Slot Variables 2 & 3																Defines Slot Variables

5.4.1 Channel *X*[1] General Settings (Words 0, 8, 16, 24)

Allows the user to enable or disable channels, set the analog output level for each channel in the event of a PLC fault, or enable HART slot variables for a given channel.

Table 5-2. Channel General Settings

To Select		Make these bit settings															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PFE (Program/Idle to Fault Enable)	Disable																0
	Enable																1
HI (Hold for Initialization)	Disable															0	
	Enable															1	
PM (Program Mode)	Hold Last State														0		
	User Defined														1		
FM (Fault Mode)	Hold Last State													0			
	User Defined													1			
ER (Enable Ramping)	Disable												0				
	Enable												1				
LA (Latch Alarms)	Disable											0					
	Enable											1					
SIO (System Interrupt High Clamp, Over Range Alarm)	Disable										0						
	Enable										1						
SIU (System Interrupt Low Clamp, Under Range Alarm)	Disable									0							
	Enable									1							
Enable Slot Variable 0	Disable								0								
	Enable								1								
Enable Slot Variable 1	Disable							0									
	Enable							1									
Enable Slot Variable 2	Disable						0										
	Enable						1										
Enable Slot Variable 3	Disable					0											
	Enable					1											
Reserved	Set to Zero		0	0	0												

[1] Where X is the channel number

To Select		Make these bit settings															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Channel Enable	Disable	0															
	Enable	1															

Real Time Sample Value (Word 0)

The real-time sample value determines when the module will scan its input channels for available data. After the channels are scanned, the data is made available to the PLC. The valid range for the real-time sample is 0 to 10000 ms (that is, Enter a value of 0 to 10000).

NOTE	The Real Time Sample rate must be greater than or equal to the slowest channel step response time. See Table 6-5 to determine the proper RTS rate.

NOTE	The configuration file can also be modified through the control program, if supported by the controller. For information on configuring the module using RSLogix 500 (with MicroLogix 1500 controller), see Chapter 5; for RSLogix 5000 (CompactLogix controller), see Chapter 4.

PFE (Program/Idle to Fault Enable)

If a system currently in program/idle mode faults, this setting determines whether the program/idle or fault value is applied to the output. If the selection is enabled [the bit is set (1)], the module applies the fault value. If the selection is disabled [the bit is reset (0)], the module applies the program/idle mode data value. The default setting is disabled.

NOTE	Not all controllers support this function. Refer to your controller's user manual for more details.

HI (Hold for Initialization)

Hold for Initialization causes outputs to hold their present state until the value commanded by the controller matches the value or is within a percentage of the value being commanded by the module. The value being commanded by the module is reflected in the Input file words 2 through 5; the channel data value words for channels 0 through 3 respectively.

If Hold for Initialization is selected, outputs will hold if any of the three conditions occur:

[8] Entering a value of zero allows the module to automatically select the fastest allowed RTS rate.

- Initial connection is established after power-up.
- Communications re-established after a communications fault.
- There is a transition from Program mode to run mode.

PM (Program Mode)

This parameter determines whether the output channel will hold the last state of the output signal or go to a user-defined value during a transition from running mode to program mode.

NOTE	The user-defined value is defined by configuration words 3, 11, 19, and 27 for channels 0 through 3 respectively.

FM (Fault Mode)

This parameter determines whether the output channel will hold the last state of the output signal or go to a user-defined value during a transition from running mode to fault mode.

NOTE	The user-defined value is defined by configuration words 2, 10, 18, and 26 for channels 0 through 3 respectively.

ER (Enable Ramping)

When enabled, the analog signal for the associated channel will increment and decrement at predefined rate whenever the channel data value changes.

LA (Latch Alarm)

The *latch alarm* feature enables process alarms (that is, Over Range, Under Range, Clamp High, etc.) to be latched during an alarm condition.

NOTE	A specific bit in the output image file must be set to clear the associated alarm latch condition. See Section 5.4 Output Data File.

SIO (System Interrupt High Clamp, Over Range Alarm)

When the SIO feature is enabled, a high clamp or over range alarm will trigger a system interrupt within the PLC. This feature allows an immediate response from the system when an analog output channel is commanded to exceed the user defined clamp limit.

SIU (System Interrupt Low Clamp, Under Range Alarm)

When the SIU feature is enabled, a low clamp or under range alarm will trigger a system interrupt within the PLC. This feature allows an immediate response from the system when an analog output channel is commanded to exceed the user defined clamp limit.

NOTE	Not all controllers support this function. Refer to your controller's user manual for more details.

SV0...SV3 (Slot Variables 0 to 3)

Enables HART slot variables 0 through 3 for the selected channel.

NOTE	Slot variables are not supported by all HART devices.

NOTE	Slot codes must be enabled in sequential order. For example, SV0 (Enabled), SV1 (Disabled), and SV2 (Enabled), is not a valid configuration. In this case, all three slot variables would be enabled.

Reserved

Must be set to Zero.

EC (Enable Channel)

This bit allows the user to enable or disable the channel.

5.4.2 Channel *X*[1] Output Type and Data Format (Words 1, 9, 17, 25)

This section of the configuration allows the user to define the output type (that is, 0 to 20 mA or 4 to 20 mA) and the data format for the associated channel. See table below.

Word 1 is used to configure general module properties like enabling and disabling HART, setting a HART handle time for HART messaging, and selecting one of three scanning schemes for HART pass-through messages. The table below shows the available settings for word 1.

Table 5-3. Output Type and Data Format

To Select			Make these bit settings															
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Output Type	0 to 20 mA															0	0	0
	4 to 20 mA															0	0	1
Reserved	Set to Zero										0	0	0	0	0			
Data Type	Raw/ Proportional							0	0	0								
	Engineering Units							0	0	1								
	Scaled for PID							0	1	0								
	Percent Range							0	1	1								
Reserved	Set To Zero		0	0	0	0	0											

Output Type

Allows the user to configure the output type and range for the associated channel.

NOTE	To enable HART, you must select the 4 to 20 mA range.

Reserved

Reserved for future expansion and should be set to zero.

Data Format

This selection allows the user to define the channel's data format for any of the following:

- **Raw/Proportional Data**
  The raw/proportional data format provides the best resolution of all the supported data formats.

  If you select the raw/proportional data format for a channel, the output command word[1] will be a number between -32767 and +32767. For example, if a 4 to 20 mA output type is selected, 4 mA corresponds to -32767 counts and 20 mA corresponds to +32767.
- **Engineering Units**
  When using this data format, the module scales the output command word[1] to the actual engineering values for the selected output type.

[1] Module output words 0 through 3 for channel 0 through 3, respectively.

Values are expressed with an assumed decimal place. Refer to Table 5-4 (Data Formats).

The resolution for the selected data format is dependent on the output type selected.

- **Scaled-for-PID**
  The PID algorithm for the MicroLogix 1500 and other Allen-Bradley controllers (that is, SLC) use an engineering range of 0 to 16383. The *scaled for PID* format allows the programmer to assign the output command word[1] for the selected channel directly to the PID instruction's CV variable, thus eliminating the need for further scaling.
- **Percent Range**
  The output command word[1] for the selected channel is scaled using a range of 0 to +10000 counts. For example, if a 4 to 20 mA output type is selected, 4 mA corresponds to 0 counts and 20 mA corresponds to +10000 counts.

NOTE	10000 counts represent 100.00 percent.

Table 5-4. Data Formats

Output Type	Engineering	Scaled for PID	Proportional Counts	Percent
0 – 20 mA	0 to 20000	0 to 16383	-32768 to 29646	0 to 10000
4 – 20 mA	4000 to 20000	0 to 16383	-29822 to 29085	0 to 10000

Reserved

Reserved for future expansion and should be set to zero.

5.4.3 Fault Value (Words 2, 10, 18, 26)

During a processor fault condition, the associated output channel will be driven to this value.

NOTE	The range of this value is defined by the data format selected for the channel.

5.4.4 Program/Idle Value (Words 3, 11, 19, 27)

During a processor program or idle mode, the associated output channel will be driven to this value.

NOTE	The range of this value is defined by the data format selected for the

	channel.

5.4.5 Low Clamp Value (Words 4, 12, 20, 28)

The low clamp value determines the minimum analog signal that will be applied to the associated channel.

NOTE	If the channel command value[1] is less than or equal to the low clamp value, the under range status bit, for the associated channel, will be set.

NOTE	The low clamp value must be less than the high clamp value or a module fault will be generated. If both the low clamp and high clamp values are left at zero, the output is clamped using the end points of the associated channel's defined output range.

5.4.6 High Clamp Value (Words 5, 13, 21, 29)

The high clamp value determines the maximum analog signal that will be applied to the associated channel.

NOTE	If the channel command value[1] is greater than or equal to the high clamp value, the over range status bit, for the associated channel, will be set.

NOTE	The low clamp value must be less than the high clamp value or a module fault will be generated. If both the low clamp and high clamp values are left at zero, the output is clamped using the end points of the associated channel's defined output range,

5.4.7 Ramp Rate Value (Words 6, 14, 22, 30)

The ramp rate value defines the rate at which the output will change state once a channel is commanded to a new value. The ramp rate is defined in terms of the selected range/format in units per second. For example, in the 0 to 20 mA DC range and percent of full scale format, a ramp rate of 1000 is 10%/second or a maximum of 1 mA DC per second. The following table describes the minimum ramp rate that can be applied for all output range/types and output data formats.

Table 5-5. Minimum Allowed Ramp Rates

[1] Module output words 0 through 3 for channel 0 through 3, respectively.

Output Data Format Output Range/Type	Total Counts in Full Scale	Number of Counts for Every 1% of Ramp Rate	Real Units/Second for Every 1% of Ramp Rate
Proportional Counts			
0 to 20 mA	65534	655	0.2 mA/Sec
4 to 20 mA	65534	655	0.16 mA/Sec
Engineering Units			
0 to 20 mA	21000	210	0.2 mA/Sec
4 to 20 mA	17800	178	0.16 mA/Sec
Scaled for PID			
0 to 20 mA	16383	164	0.2 mA/Sec
4 to 20 mA	16383	164	0.16 mA/Sec
Percent of Full Scale			
0 to 20 mA	10000	100	0.2 mA/Sec
4 to 20 mA	10000	100	0.16 mA/Sec

NOTE	If you enter a ramp rate value that is less than the minimum specified in the table above, a module fault will be generated.

NOTE	The number of counts entered for a channel's ramp rate may be equal to 0 only if ramping is not enabled for the channel

5.4.8 Spare (Words 7, 15, 23, 31)

Reserved for future expansion.

5.4.9 Pad (Word 32)

1-bit alignment (Set to Zero)

5.4.10 HART Configuration Word (Word 33)

Word 33 is used to configure module properties like enabling and disabling HART, setting a HART handle time for HART messaging, and selecting one of three scanning schemes for HART pass-through messages. The table below shows the available settings for word 33.

To Select		Make these bit settings															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Handle Timeout										Handle Timeout (1 to 255 sec)							
CH0 HART Enable	Disable								0								
	Enabled								1								
CH1 HART Enable	Disable							0									
	Enabled							1									
CH2 HART Enable	Disable						0										
	Enabled						1										
CH3 HART Enable	Disable					0											
	Enabled					1											
Pass-Through Scheme	Two Channel Scans			0	0												
	Once Per Module Scan			0	1												
	Every Channel Scan			1	0												
Reserved	Set to Zero	0	0														

Handle Timeout

There is a handle timeout associated with the final reply message. After the module obtains the requested information from the HART device, it will start the Handle Timeout timer. The reply message will be kept in memory during the Handle Timeout period. After the timeout occurs or after the message is retrieved by the pass-through response query command, the storage buffer will be discarded, and another pass through message will be serviced without being rejected. Handle Timeout is in the range of 0 to 255 seconds.

NOTE	A handle timeout of zero is valid. When set to zero, the handle timeout will default to 10 seconds.

Channel HART Enable (Bits 8, 9, 10, 11)

These bits allow the user to enable HART on channels 0 through 3, respectively.

Pass-Through Scheme

The pass-through scheme determines how often the pass through commands are serviced.

- Two Channel Scans: Pass-through serviced once every two channel scans.

- Once Per Module Scan: Pass-through serviced once per module scan.
- Every Channel Scan: Pass-through serviced once every channel scan.

NOTE	The pass-through scheme can increase the HART packet update time if pass-through messages are serviced every channel scan.

Enable HART Logging

Enables packet five which allows the HART logging to take place. The HART logging function allows the user to view and verify the execution of HART commands that are sent to each channel during the HART acquisition cycle. This feature should only be used if you're experiencing HART communication problems. For more details, please refer Chapter 5.

5.4.11 Channel *X*[1] HART Slot Variables 0 & 1 (Words 34, 36, 38, 40)

This word defines HART slot variables 0 and 1 for the selected channel. The first byte defines slot variable 0 and the second defines slot variable 1. The variable is defined as a hexadecimal value between 0 and FF.

The HART slot variable is a floating point value that represents a device-specific variable defined by the manufacturer for the connected HART field device. This is an optional configuration setting and is not supported by all HART field devices. For more information regarding slot variables, refer to Section 6.2.

5.4.12 Channel *X*[1] HART Slot Variables 2 & 3 (Words 35, 37, 39, 41)

This word defines HART slot variables 2 and 3 for the selected channel. The first byte defines slot variable 2 and the second defines slot variable 3.

The HART slot variable is a floating-point value that represents a device specific variable defined by the manufacturer for the connected HART field device. This is an optional configuration setting and is not supported by all HART field devices.

For more information regarding slot variables, refer to Section 6.2.

Section 5.5 Output Data File

The output data file is used to command the analog signal for each channel as well as controlling features which include clearing process alarms, suspending HART acquisition, and managing HART messages to, and from, HART field devices. The data table structure is shown in the table below.

[1] Where X is the channel number (0 to 3)

Table 5-6. Output Data File

Word/ Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Channel 0 Command Value															
1	Channel 1 Command Value															
2	Channel 2 Command Value															
3	Channel 3 Command Value															
4	HS3	HS2	HS1	HS0	Reserved				UL3	UH3	UL2	UH2	UL1	UH1	UL0	UH0
5	Packet Just Scanned															
6	Message Master Control															
7	Message Request Size															
8..27	Message Request Buffer															
28..49	Reserved															

5.5.1 Channel *X*[1] Command Value

The command value is used to control the analog output signal for each channel. The command value is entered in engineering units defined by the channel's selected data format and output type. Refer to section 5.4.2 for more details regarding data format and output type.

5.5.2 Unlatch Process High Alarms UH0 to UH3 (Word 4)

UH0 through UH3 will unlatch the high process alarms for channels 0 through 3 respectively. To unlatch the high process alarm on a given channel, set the unlatch bit to 1.

NOTE	Setting the unlatch process alarm bit will not clear the alarm latch if the conditions that generated the alarm are still present.

NOTE	It is up to the user to keep the unlatch bit set until verification that the process alarm bit has cleared. When the process alarm bit has cleared the user can then clear the unlatch process alarm bit.

NOTE	The module will not latch the high process alarm if a transition from "no alarm condition" to "alarm condition" occurs while the unlatch high process alarm bit is set.

5.5.3 Unlatch Process Low Alarms UL0 to UL3 (Word 4)

UL0 through UL3 will unlatch the low process alarms for channels 0 through 3 respectively. Refer to section 0 for more information regarding setting the alarm latch function. To unlatch the low process alarm on a given channel, set the associated unlatch alarm bit to 1.

NOTE	Setting the unlatch process alarm bit will not clear the alarm latch if the conditions that generated the alarm are still present.

NOTE	It is up to the user to keep the unlatch bit set until verification that the process alarm bit has cleared. When the process alarm bit has cleared, the user can then clear the unlatch process alarm bit.

NOTE	The module will not latch the low process alarm if a transition from "no alarm condition" to "alarm condition" occurs while the unlatch low process alarm bit is set.

5.5.4 Hart Suspend HS0 to HS3 (Word 4)

HS0 to HS3 are used to suspend all HART acquisition, except Pass-through messages, on channels 0 through 3 respectively. To suspend HART acquisition, set the associated channel suspend bit to 1. Normal HART acquisition will resume when the bit is cleared.

NOTE	A channel that is placed out-of-service (that is, Suspended) will automatically resume service after three minutes, as long as no pass-through commands are issued before the three minutes expires.

5.5.5 Packet Just Scanned (Word 5)

When demultiplexing HART data from the module, this output word can be used to speed up the acquisition process by overriding the automatic 500 ms acquisition delay between packets.[1] To override the delay, the *packet just scanned* word needs to be populated with word six from the input data file on each scan of the ladder program. Input word six contains the channel and packet number just scanned.

NOTE	Input word six is the first word of twenty which contains the multiplexed HART data for each channel.

5.5.6 Message Master Control (Word 6)

This word is used to control the data flow of a message sent to the module.

[1] Refer to Chapter 7 for more details.

These messages include module commands such as HART pass-through, HART suspend and resume, and get device information.[1]

5.5.7 Message Request Size (Word 7)

The *message request size* determines the size of the message, in bytes, that will be sent to the module.[1]

5.5.8 Message Request Buffer (Words 8…27)

The *message request buffer* contains the data making up the message that will be sent to the module.[1]

5.5.9 Reserved (Words 28…49)

Reserved for future expansion.

Section 5.6 Determining Module Update Time

The module update time is defined as the time required for the module to receive the command value and convert it to an equivalent analog output signal for all enabled channels.

5.6.1 Module Update Time

HART Enabled	HART Disabled
12 ms	10 ms

Chapter 6
Enabling and Using HART on the 1769sc-OF4IH

This chapter outlines the detailed settings and configuration related to HART communication for the 1769sc-OF4IH module. These settings determine how the module acquires HART data.

The chapter is broken down into the following sections:

- Configuring the module for HART
- HART Packet Data
- Sending and Receiving Messages
- Module Specific Commands
- HART protocol overview

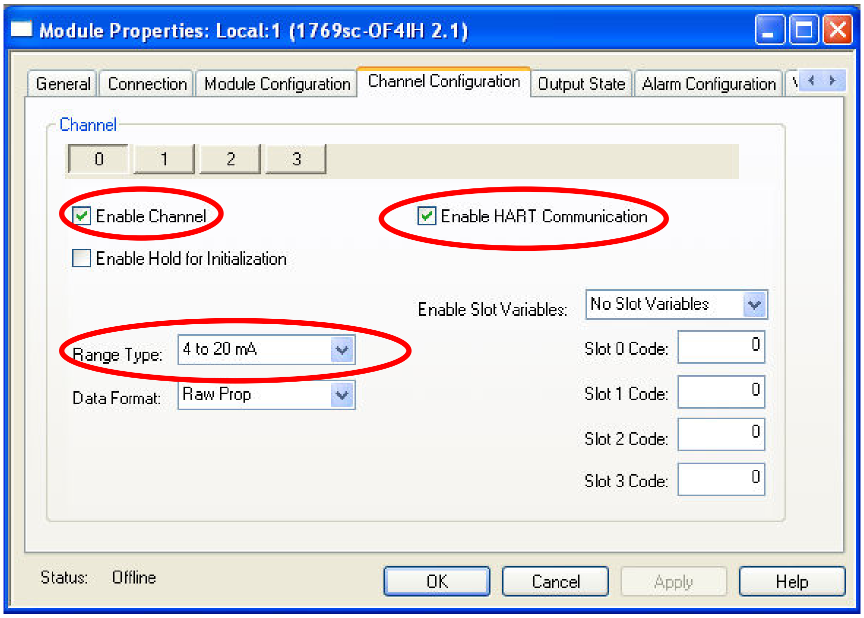
NOTE	The ladder samples and tags referenced in this chapter were created for the Compact Logix controller using RSLogix 5000 software, see Chapter 4. If you plan on using a MicroLogix 1500 controller, refer to Chapter 5.

Section 6.1 Configuring the Module for HART

6.1.1 Configuring the OF4IH Module for (Hart Acquisition/Communication)

For HART to be active on any given channel, the channel configuration must contain the following basic settings.

The channel must be enabled, set for 4 to 20 mA, and the **Enable HART Communication** checkbox must be checked.



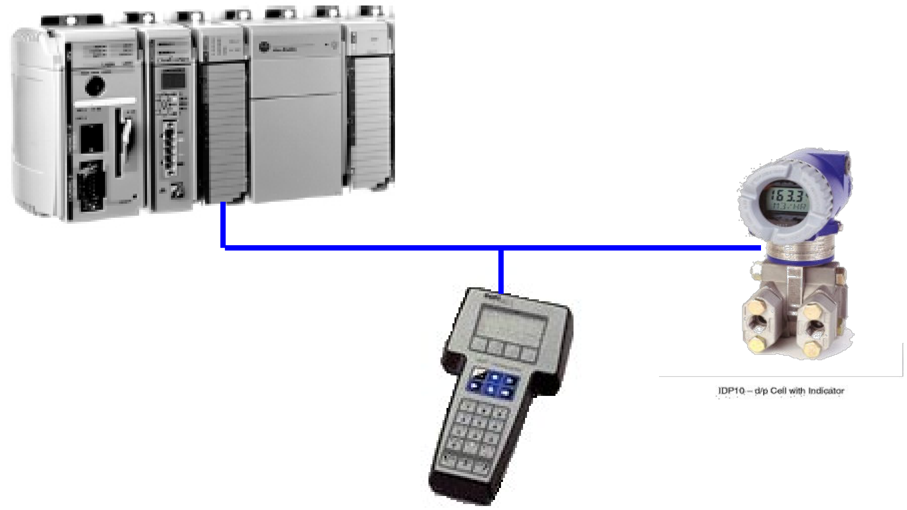
NOTE	HART throughput time can be improved by disabling HART communication on unused channels or channels that include non-HART devices.

Section 6.2 HART Packet Data

6.2.1 How the Module Connects to a Field Device

The HART output module behaves as a HART master in which case the field device is considered the slave. In other words, the master must initiate the communication with the field device, and the device simply replies with an appropriate response. Any given channel may have a master, a secondary master (hand-held configuration tool), and a slave connected simultaneously. Please see Figure below.

Figure 6-1. System Layout Example



NOTE	HART multi-drop is not supported by the OF4IH.

The HART module communicates to the controller using the input and output image. Data communicated over the input and output image are transmitted at a rate that is controlled by the PLC. The rate at which data is communicated to the CompactBus is adjustable by using the RPI (Requested Packet Interval). The data passed via the input and output image include, analog data, module status, HART data, and module specific commands.

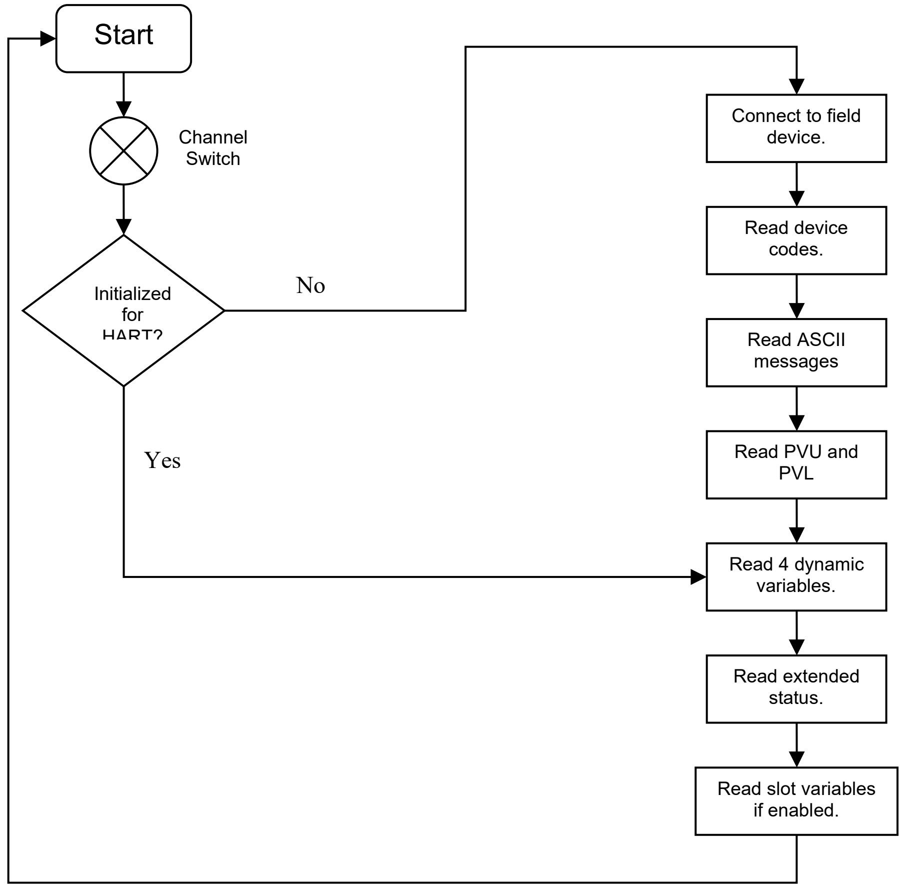
Module specific commands include the HART pass-through commands, HART suspend, HART resume, and the get HART device information command.

Gathering HART data is accomplished using two processes **auto acquisition**, and or using the **module specific commands**.

6.2.2 Auto Acquisition

When a channel is configured for HART, the module will automatically search and establish a connection to any HART field device wired to the channel. Once the module establishes a connection it will begin to acquire HART data, including device specific codes (that is, Manufacturer ID, serial number, etc.), the four dynamic variables, extended device status, slot variables (if enabled), and any stored ASCII message descriptor that may be present. The HART data retrieved automatically by the module is then displayed in the input image (Of4ih0Input.HartData) and is accessible by ladder logic. The HART data will update, on average, every 3.5 seconds if all four channels are enabled for HART. The module initiates the connection by sending a string of HART commands to the field device. Please see figure below.

Figure 6-2. Auto Acquisition Flow



The data that is collected from the process described in Figure 7-4 (Auto Acquisition Flow) is buffered to the module RAM memory. Since the amount of data returned from the auto-acquisition process is extensive, the data is multiplexed into five separate packets and for each individual channel. The multiplexed data can be read from a 40-byte array that is located in the OF4IH0Input.HartData tag. The multiplexed data is demultiplexed using ladder and stored in five different arrays which are structured using packets zero through four. The packets are defined as “user defined data types” and can be seen in Table 6-1 through Table 6-5.

Table 6-1. HART Packet 0

Tag Name	Data Type	Style	Description
If4ih0Packet0	Packet0[4,1]	NA	Two-dimensional array containing packet 0 data for all 4 channels.
If4ih0Packet0[X,0][17]	Packet0	NA	Packet 0 data for channel *X*.
If4ih0Packet0[X,0].HartChannelID	INT	BIN	Bits 0 to 3: Channel number (0 – 3). Bit 4: Searching/Initializing HART device Bit 5: HART communication failure or device not found Bit 6: Pass-through message pending (ready) Bit 7: Unused (0) Bits 8 to 10: Packet ID Bit 11 through 15: Unused
If4ih0Packet0[X,0].ManufacturerID	SINT	DEC	HART device Manufacturer ID
If4ih0Packet0[X,0].DeviceType	SINT	DEC	HART device type code
If4ih0Packet0[X,0].NumPreambles	SINT	DEC	Minimum number of preambles the device requires.
If4ih0Packet0[X,0].UniversalCmdCode	SINT	DEC	HART Universal command set 5.0
If4ih0Packet0[X,0].XmitterRev	SINT	DEC	HART Transmitter specific revision
If4ih0Packet0[X,0].SwRev	SINT	DEC	HART device software revision number
If4ih0Packet0[X,0].HwRev	SINT	DEC	HART device hardware revision number
If4ih0Packet0[X,0].HartFlags	SINT	BIN	HART flags
If4ih0Packet0[X,0].RangeUnits	SINT	DEC	Units code for range parameter
If4ih0Packet0[X,0].DeviceSerialNumber	SINT[3]	HEX	HART device ID number
If4ih0Packet0[X,0].DeviceTag	SINT[8]	ASCII	8-character device tag
If4ih0Packet0[X,0].DeviceDescriptor	SINT[16]	ASCII	

[17] *X* represents the module channel number (0 to 3)

Table 6-2. HART Packet 1

Tag Name	Data Type	Style	Description
If4ih0Packet1[1]	Packet1[4,1]	NA	Two-dimensional array containing packet 1 data for all 4 channels.
If4ih0Packet1[X,0] [18]	Packet1	NA	Packet 1 data for channel *X*
If4ih0Packet1[X,0].HartChannelID	INT	BIN	Bits 0 to 3: Channel number (0 – 3). Bit 4: Searching/Initializing HART device Bit 5: HART communication failure or device not found Bit 6: Pass-through message pending (ready) Bit 7: Unused (0) Bits 8 to 10: Packet ID Bit 11 through 15: Unused
If4ih0Packet1[X,0].HartCommStatus	SINT	BIN	HART communication status byte. Refer to appendix B for more details.
If4ih0Packet1[X,0].HartDevStatus	SINT	BIN	HART device status byte. Refer to appendix B for more details.
If4ih0Packet1[X,0].HartPV	REAL	FLOAT	HART Primary Variable
If4ih0Packet1[X,0].HartSV	REAL	FLOAT	HART Secondary Variable
If4ih0Packet1[X,0].HartTV	REAL	FLOAT	HART Tertiary Variable
If4ih0Packet1[X,0].HartFV	REAL	FLOAT	HART Fourth Variable
If4ih0Packet1[X,0].HartPVUnits	SINT	DEC	HART Primary Variable units code
If4ih0Packet1[X,0].HartSVUnits	SINT	DEC	HART Secondary Variable units code
If4ih0Packet1[X,0].HartTVUnits	SINT	DEC	HART Tertiary Variable units code
If4ih0Packet1[X,0].HartFVUnits	SINT	DEC	HART Fourth Variable units code
If4ih0Packet1[X,0].PV_Assignment	SINT	DEC	HART Primary Variable code
If4ih0Packet1[X,0].SV_Assignment	SINT	DEC	HART Secondary Variable code

[18] *X* represents the module channel number (0 to 3)

Tag Name	Data Type	Style	Description
If4ih0Packet1[X,0].TV_Assignment	SINT	DEC	HART Tertiary Variable code
If4ih0Packet1[X,0].FV_Assignment	SINT	DEC	HART Fourth Variable code
If4ih0Packet1[X,0].RangeLow	REAL	FLOAT	Low transmitter range for analog signal in engineering units
If4ih0Packet1[X,0].RangeHi	REAL	FLOAT	High transmitter range for analog signal in engineering units
If4ih0Packet1[X,0].Pad	SINT[4]	DEC	Packet pad (32 bit alignment)

Table 6-3. HART Packet 2

Tag Name	Data Type	Style	Description
If4ih0Packet2	Packet2[4,1]	NA	Two-dimensional array containing packet 2 data for all 4 channels.
If4ih0Packet2[X,0] [19]	Packet2	NA	Packet 2 data for channel *X*
If4ih0Packet2[X,0].HartChannelID	INT	BIN	Bits 0 to 3: Channel number (0 – 3). Bit 4: Searching/Initializing HART device Bit 5: HART communication failure or device not found Bit 6: Pass-through message pending (ready) Bit 7: Unused (0) Bits 8 to 10: Packet ID Bit 11 through 15: Unused
If4ih0Packet2[X,0].Slot0Data	REAL	Float	Variable for slot 0
If4ih0Packet2[X,0].Slot1Data	REAL	Float	Variable for slot 1
If4ih0Packet2[X,0].Slot2Data	REAL	Float	Variable for slot 2
If4ih0Packet2[X,0].Slot3Data	REAL	Float	Variable for slot 3
If4ih0Packet2[X,0].Slot0Units	SINT	DEC	Slot 0 units code
If4ih0Packet2[X,0].Slot1Units	SINT	DEC	Slot 1 units code
If4ih0Packet2[X,0].Slot2Units	SINT	DEC	Slot 2 units code
If4ih0Packet2[X,0].Slot3Units	SINT	DEC	Slot 3 units code
If4ih0Packet2[X,0].Slot0Assignment	SINT	DEC	Slot 0 variable code
If4ih0Packet2[X,0].Slot1Assignment	SINT	DEC	Slot 1 variable code

[19] *X* represents the module channel number (0 to 3)

Tag Name	Data Type	Style	Description
If4ih0Packet2[X,0].Slot2Assignment	SINT	DEC	Slot 2 variable code
If4ih0Packet2[X,0].Slot3Assignment	SINT	DEC	Slot 3 variable code
If4ih0Packet2[X,0].Pad	SINT[12]	DEC	Packet pad

Table 6-4. HART Packet 3

Tag Name	Data Type	Style	Description
If4ih0Packet3	Packet3[4,1]	NA	Two-dimensional array containing packet 3 data for all 4 channels.
If4ih0Packet3[X,0][1]	Packet3	NA	Packet 3 data for channel *X*
If4ih0Packet3[X,0].HartChannelID	INT	BIN	Bits 0 to 3: Channel number (0 – 3). Bit 4: Searching/Initializing HART device Bit 5: HART communication failure or device not found Bit 6: Pass-through message pending (ready) Bit 7: Unused (0) Bits 8 to 10: Packet ID Bit 11 through 15: Unused
If4ih0Packet3[X,0].Message	SINT[32]	ASCII	32-character message
If4ih0Packet3[X,0].Pad	SINT[4]	DEC	Pad 32-bit alignment.

Table 6-5. HART Packet 4

Tag Name	Data Type	Style	Description
If4ih0Packet4	Packet4[4,1]	NA	Two-dimensional array containing packet 4 data for all 4 channels.
If4ih0Packet4[X,0].HartChannelID	INT	BIN	Bits 0 to 3: Channel number (0 – 3). Bit 4: Searching/Initializing HART device Bit 5: HART communication failure or device not found Bit 6: Pass-through message pending (ready) Bit 7: Unused (0) Bits 8 to 10: Packet ID Bit 11 through 15: Unused
If4ih0Packet4[X,0].Date	SINT[3]	DEC	Stored date in the field device
If4ih0Packet4[X,0].FinalAssemblyNumber	SINT[3]	DEC	The final assembly number is used for identifying the materials and electronics that comprise the field device.

Tag Name	Data Type	Style	Description
If4ih0Packet4[X,0].ExtendedStatus	SINT[24]	DEC	The extended status returned by HART command 48
If4ih0Packet4[X,0].Pad	SINT[3]	DEC	Pad 32-bit alignment

NOTE	Not all the HART data that is returned by the process outlined in Figures 6-3/6-4 (Auto Acquisition Flow) gets passed to the packets. To access the data that is not passed to the packets, you must execute the appropriate HART message using the pass-through command, which will be discussed later in this chapter.

The ladder determines which packet to copy the data to, by monitoring the state of bits 0, 1, 2 and 8, 9, 10, found in the first two bytes of the OF4IH0Input.HartData tag. Bits 0, 1, 2 determine the current channel being scanned and bits 8, 9, and 10 determine the packet number. The ladder example, shown in the figure, performs this operation.

Figure 6-3. Packet Ladder

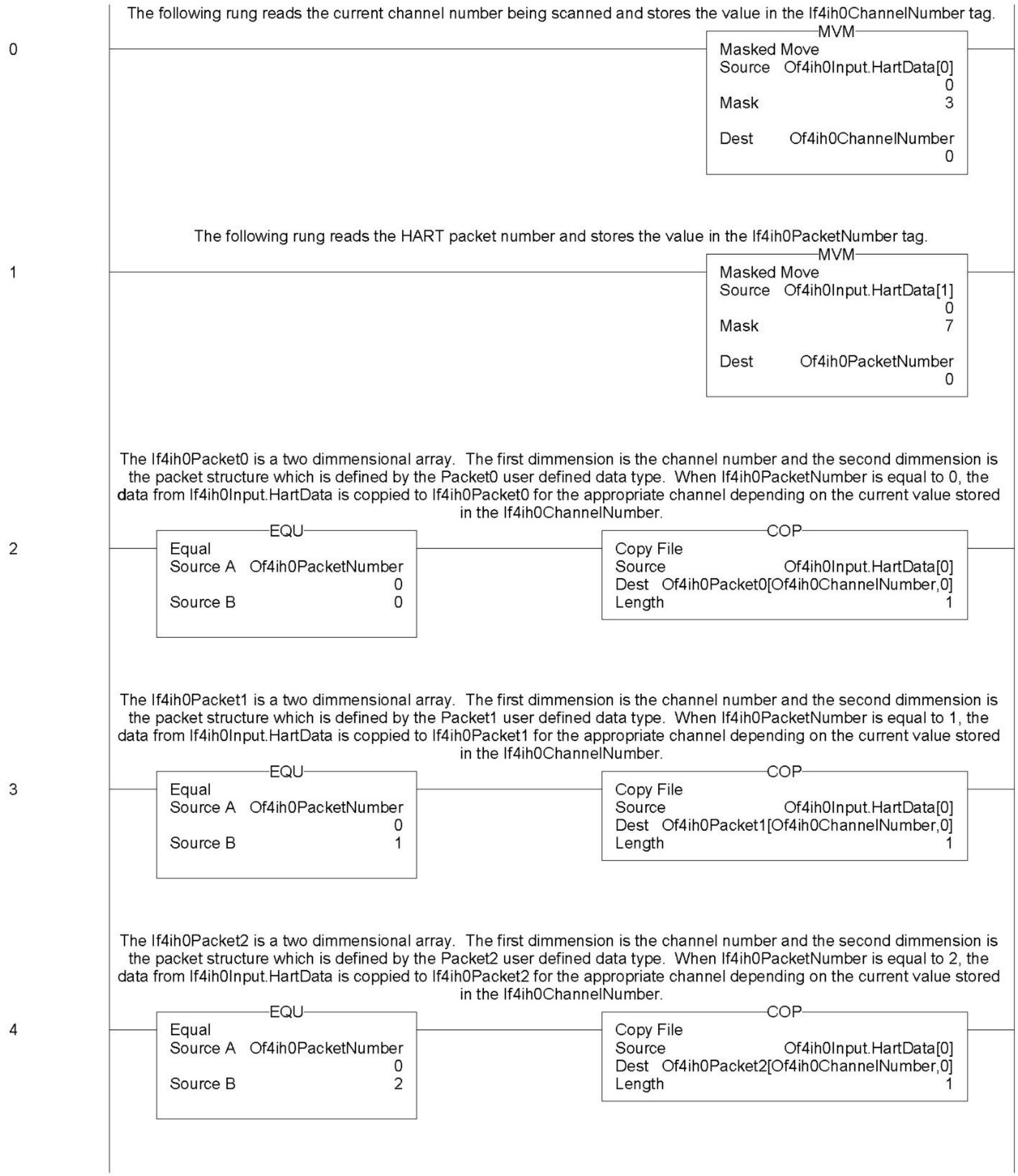
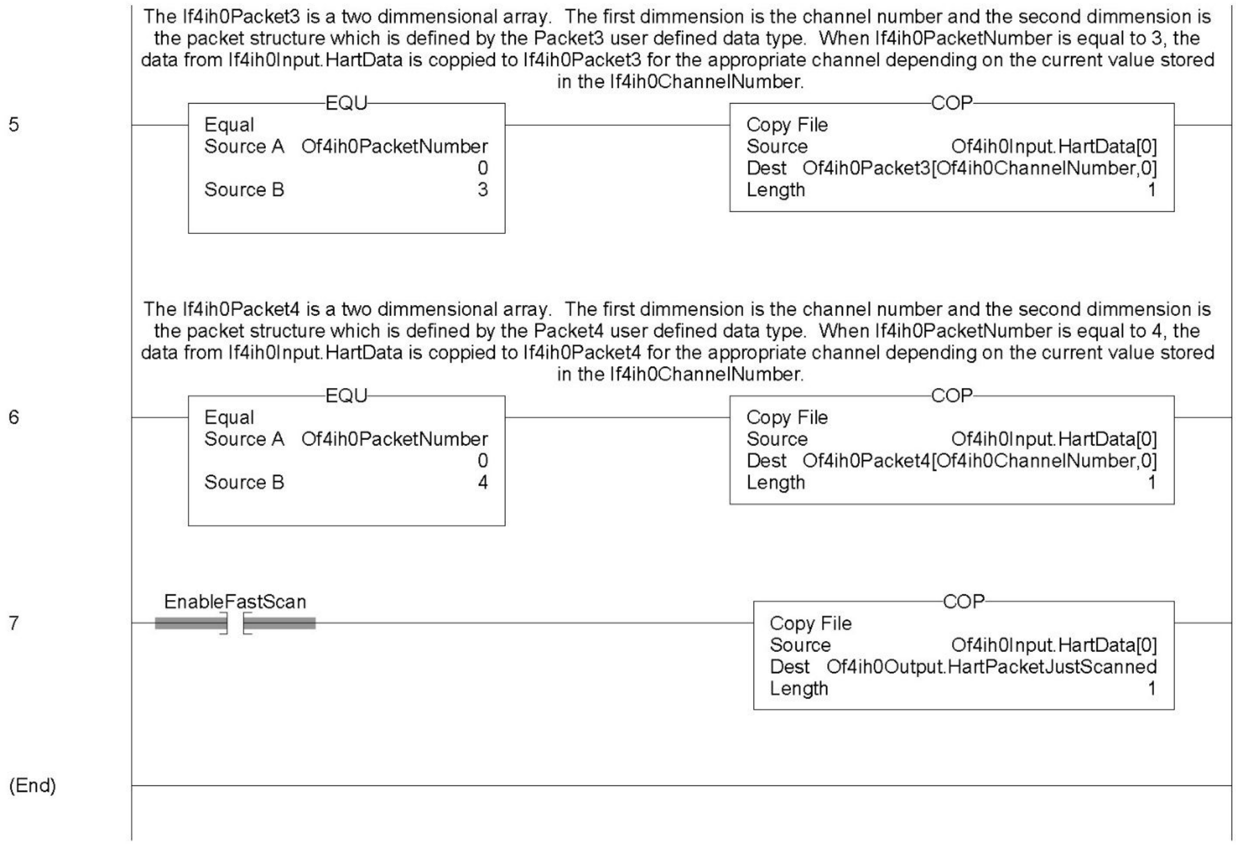


Figure 6-4. Packet Ladder Continued



NOTE	The ladder in Figures 6-3/6-4 can be found in the project sample file located on our website at www.spectrumcontrols.com

6.2.3 Packet Interval

The delay between two consecutive packets is called the packet interval. The default time for the packet interval is 500 ms. This delay is controlled by the module.

You may reduce the packet interval by using output word 1 (*HART Packet Just Scanned*) in the output image. See Table 6-6 (Module Output Table). Copying the packet number just scanned to output word 1 allows the module to switch to the next packet before the 500 ms delay expires. See Figure 6-5.

NOTE	The amount of time saved using this method depends on the scan time of the ladder and the update time of each individual HART transmitter.

Section 6.3 Sending and Receiving Messages

Sending messages to and from the module is accomplished using a paging scheme. This paging scheme uses the module's input and output words to transfer data between the controller and the module, 38 bytes at a time (that is, one page at a time). The paging scheme is used to minimize the number of bytes sent and received at one time from the module's input and output image. The maximum message size is 257 bytes.

6.3.1 Module Output Tags Used for Messaging

The OF4IH module uses 28 words for sending messages and controlling data flow. The table below shows the output image for the OF4IH module. For more detail regarding module output image, refer to Section 5.5.

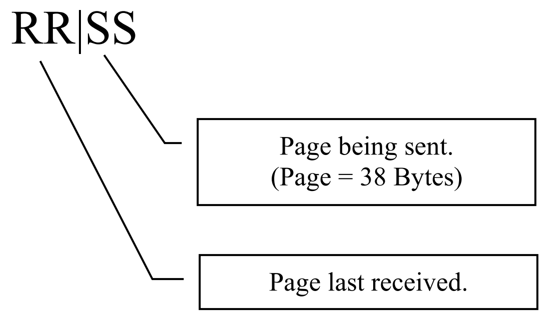
Table 6-6. Module Output Table

Word/ Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Channel 0 Command Value															
1	Channel 1 Command Value															
2	Channel 2 Command Value															
3	Channel 3 Command Value															
4	HS3	HS2	HS1	HS0	Reserved				UL3	UH3	UL2	UH2	UL1	UH1	UL0	UH0
5	Packet Just Scanned															
6	Message Master Control															
7	Message Request Size															
8..27	Message Request Buffer															
28..49	Reserved															

Word 6 (Message Master Control)

The *message master control* initiates the paging process and controls the flow of data to and from the module. The data flow control is accomplished by using the *message master control* with the *message slave control* to manage which pages are being sent and what direction the page is going, that is, whether the page is being sent to the module or read from the module.

Figure 6-5. Message Master/Slave Control (Hex)



NOTE	Setting the Message Master Control word to zero resets the paging logic within the module and allows the next message to be processed.

Word 7 (Message Request Size)

The message request size is the total number of bytes being sent to the module (not just the current page).

Words 8…27 (Message Request Buffer)

The message request buffer contains the data being sent to the module for the current page (up to 38 bytes).

6.3.2 Module Input Tags Used For Messaging

The module uses 48 input words to receive messages and control data flow. The table below shows the input words used by the module. Refer to Chapter 5 for more information regarding input words 0 through 48.

Table 6-7. Module Input Table

Word/Bit[20]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	OS3	OS2	OS1	OS0	Not Used								S3	S2	S1	S0
1	0	H3	U3	O3	0	H2	U2	O2	0	H1	U1	O1	0	H0	U0	O0
2	Channel 0 Command Value															
3	Channel 1 Command Value															
4	Channel 2 Command Value															
5	Channel 3 Command Value															
6..25	HART Packet Data															
26	Message Slave Control															
27	Message Response Size															
28..47	Message Response Buffer															
48..69	Reserved															

Word 26 (Message Slave Control)

Again, the *message slave control* is used with the *message master control* to manage which pages are being sent and what direction the page is going, that is, whether the page is being sent to the module or read from the module. Refer to Figure 6-1 for the layout. The *message slave control* is also used to indicate if a message was rejected by the module. If a message is rejected, the lower 8 bits will be set (that is, FF Hex) in the *message slave* control. If the message is rejected, the *message response buffer* will display a fault code in the first byte followed by a checksum in the second. The table below lists the possible responses:

Table 6-8. Paging Error Codes

Error Code	Description
1	A page was sent out of sequence.
2	While processing page 2, 3, etc. The message size was different than it was for page 1.
3	The message size given exceeds the max allowed.
4	The message page data checksum is not correct.

Word 27 (Message Response Size)

The *message response size* indicates the total number of bytes being returned by the module.

Words 28…47 (Message Response Buffer)

The *message response buffer* contains the response data for the current page up to thirty-eight bytes at a time.

[20] Changing bit values is not supported by all controllers. Refer to your controller manual for details.

6.3.3 Processing a Message

To complete a message from beginning to end, follow the steps listed below:

1. Store the message you wish to send to the module in an array. Remember the message can be up to 257 bytes long, so make the array large enough.
2. Copy the first page of data, up to 38 bytes, to the *message request buffer*. If the number of bytes is odd, the last byte in the last word will be padded with a zero.
3. Calculate the checksum of the message by taking the exclusive OR of all the words within the page (19 max). Place the result into the last word of the message (that is, word # 20 if a full page).
4. Enter the size of the message to be sent to the module into the *message request size* output word.
5. Add a 1 to the lower nibble of the *message master control* word (that is, 0001Hex). The *message master control* should be zero when the message is started.
6. Wait for the module to reply that it has received the page without error, by monitoring the second nibble of the *message slave control* (that is, 0100). If the lower nibble contains FF, stop the process because the data is corrupted. The first byte in the *message response buffer* will contain the paging error code. Refer to Table 6-1 for a description of the errors.
7. Check to see if there are more pages to send by comparing the bytes sent to the *message request size*. If so, repeat steps 2 through 6. If not, go to step 8.
8. Monitor the lower nibble of the *message slave control* to see if the first page of the response data is ready (0101).
9. Copy the first page of the response data from the *message response buffer* to a temporary array.
10. Take the exclusive OR of all the words within the page (19 max) with the exception of the last word which is the checksum. Compare the calculated checksum with the checksum stored in the last byte. If they are equal, go to step 11. If they are not, stop the process because the data is corrupted.
11. Check to see if there is more response data remaining by comparing the bytes received to the *message response size*. If so, repeat steps 8 through 10. If not, the message is finished. To send another message clear the *message master control* and repeat the process.

A graphical representation of the process can be seen in Figures 6-2 and 6-3.

Figure 6-6. Sending Message

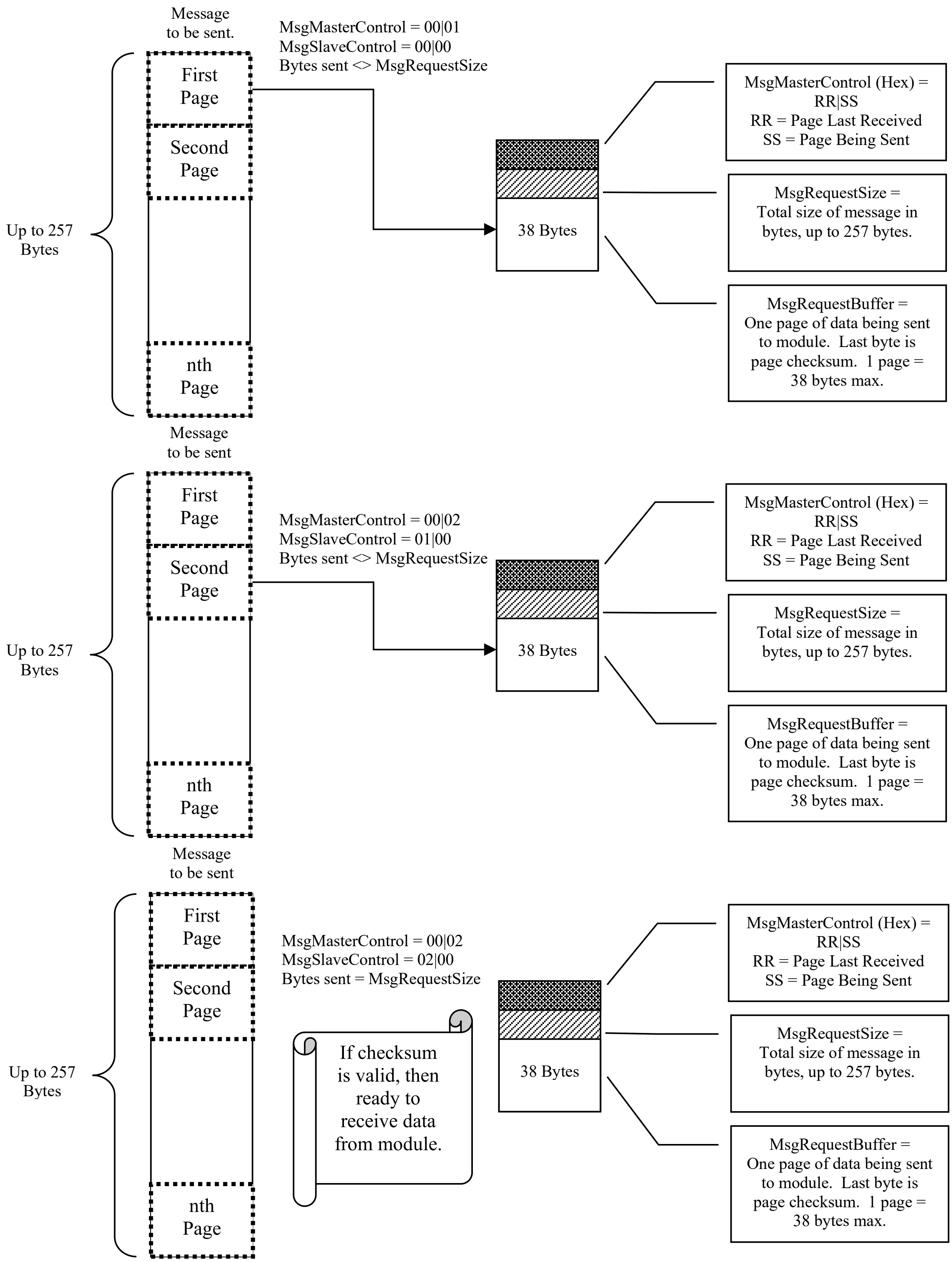


Figure 6-7. Receiving Message

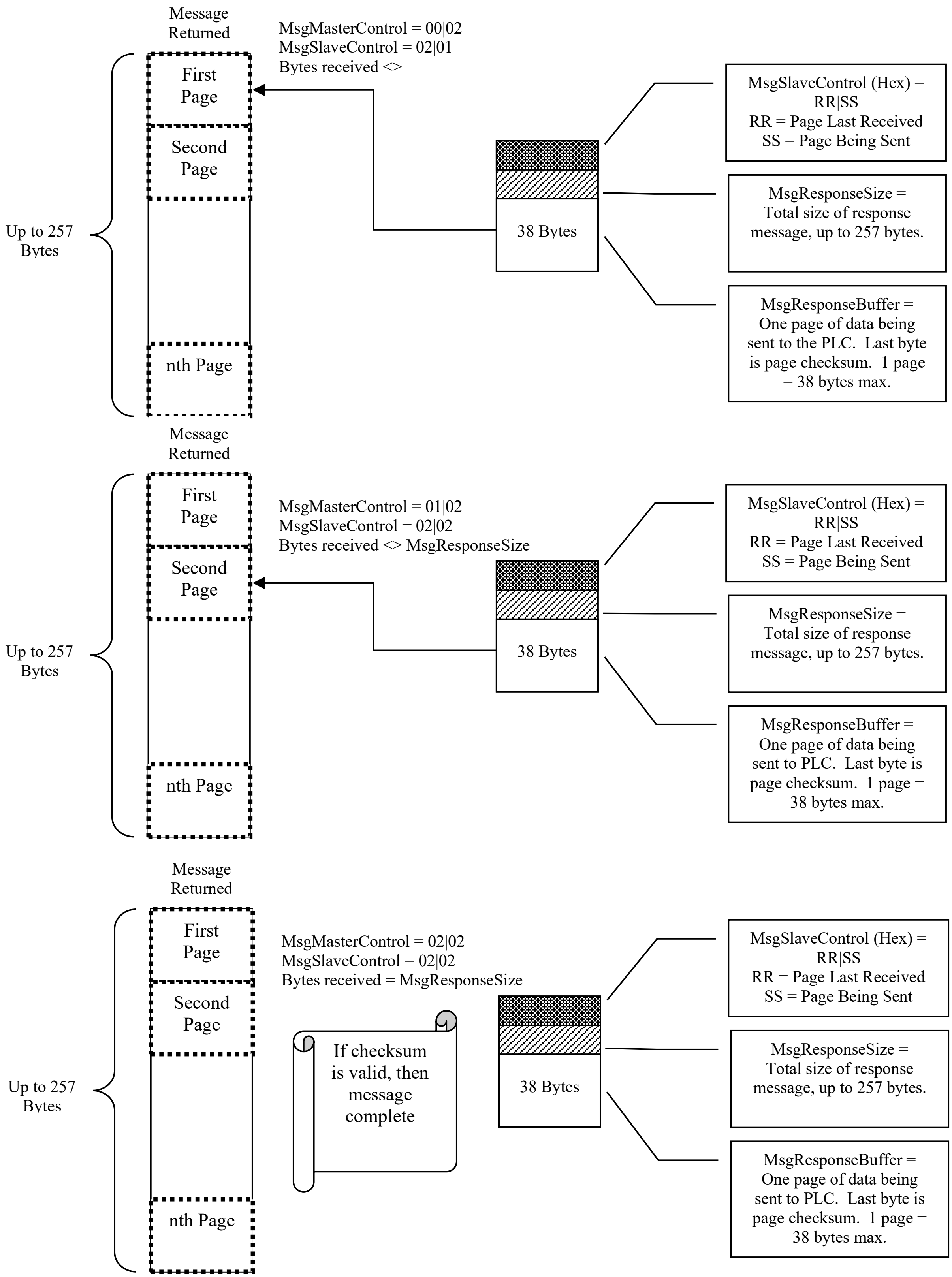


Figure 6-8a. Message Ladder

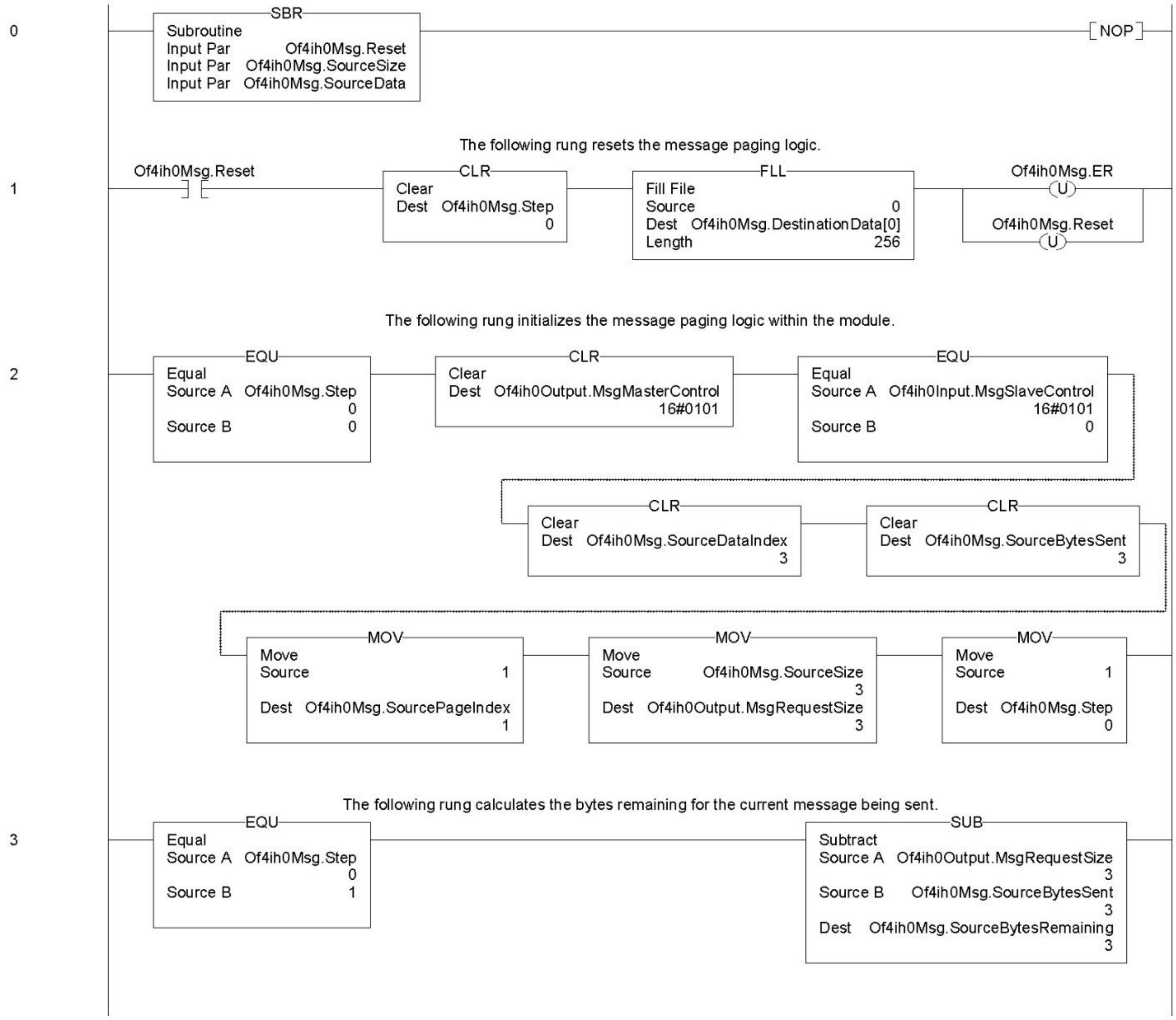


Figure 6-8b.

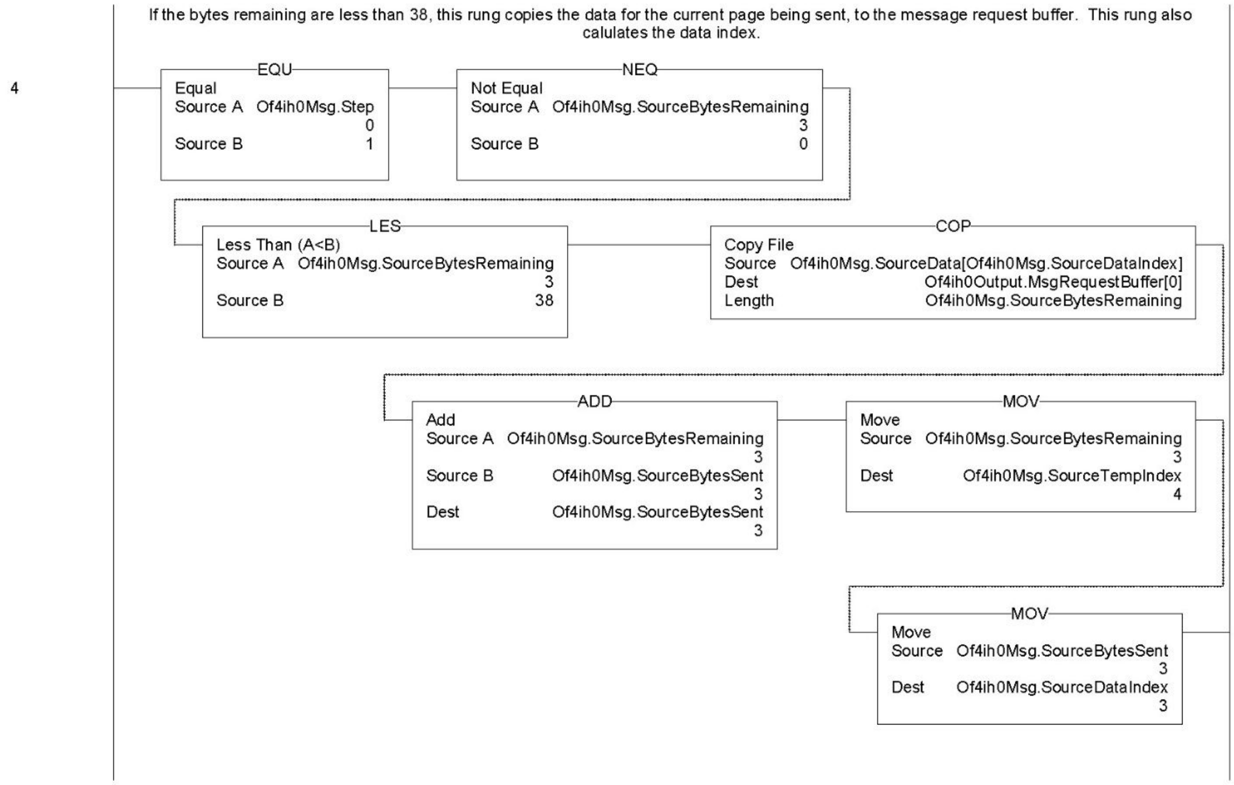

If the bytes remaining are less than 38, this rung copies the data for the current page being sent, to the message request buffer. This rung also calulates the data index.
4
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 1
NEQ
Not Equal
Source A Of4ih0Msg.SourceBytesRemaining
3
Source B 0
LES
Less Than (A<B)
Source A Of4ih0Msg.SourceBytesRemaining
3
Source B 38
COP
Copy File
Source Of4ih0Msg.SourceData[Of4ih0Msg.SourceDataIndex]
Dest Of4ih0Output.MsgRequestBuffer[0]
Length Of4ih0Msg.SourceBytesRemaining
ADD
Add
Source A Of4ih0Msg.SourceBytesRemaining
3
Source B Of4ih0Msg.SourceBytesSent
3
Dest Of4ih0Msg.SourceBytesSent
3
MOV
Move
Source Of4ih0Msg.SourceBytesRemaining
3
Dest Of4ih0Msg.SourceTempIndex
4
MOV
Move
Source Of4ih0Msg.SourceBytesSent
3
Dest Of4ih0Msg.SourceDataIndex
3

Figure 6-8c.

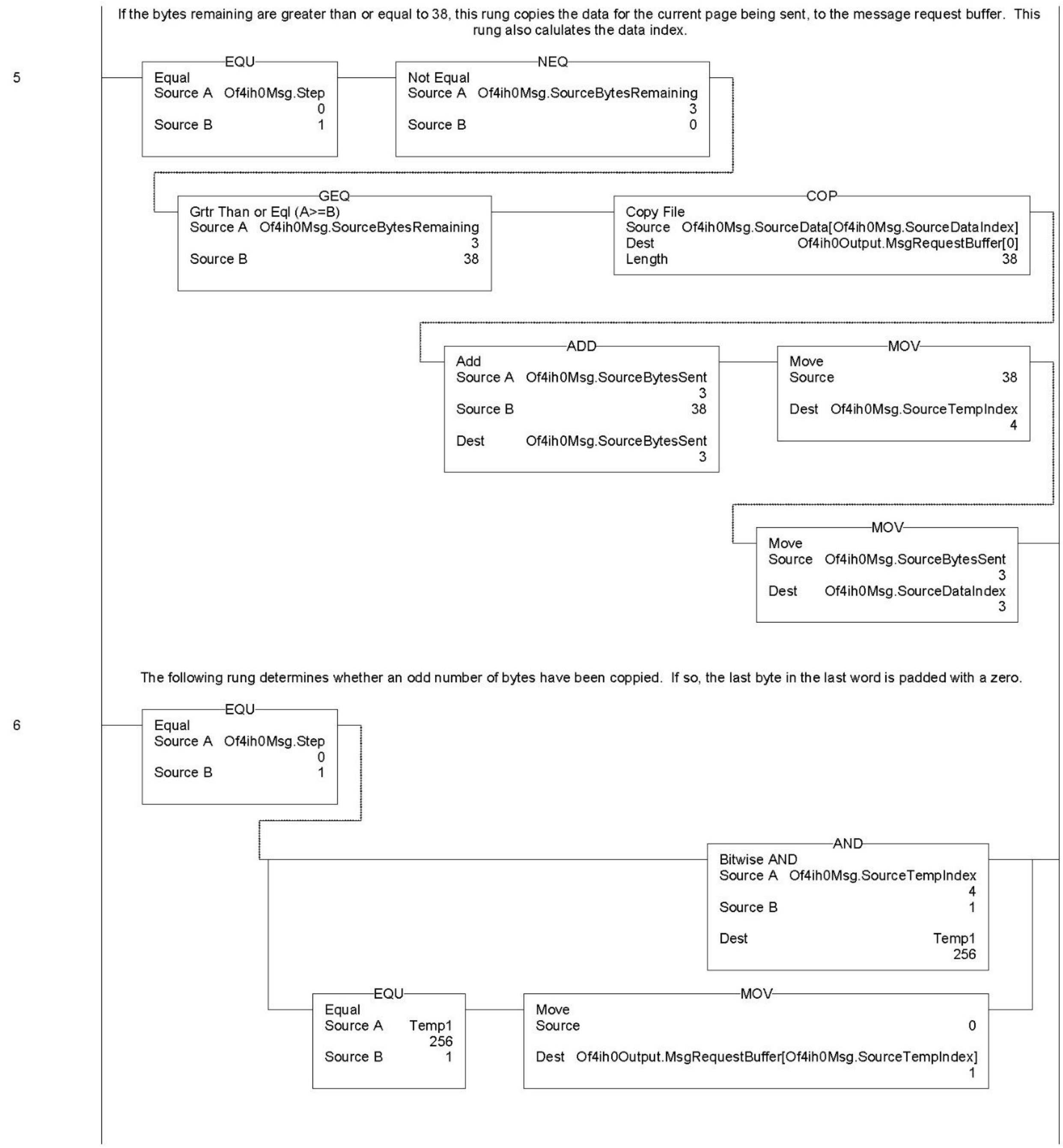

If the bytes remaining are greater than or equal to 38, this rung copies the data for the current page being sent, to the message request buffer. This rung also calulates the data index.
5
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 1
NEQ
Not Equal
Source A Of4ih0Msg.SourceBytesRemaining
3
Source B 0
GEQ
Grtr Than or Eql (A>=B)
Source A Of4ih0Msg.SourceBytesRemaining
3
Source B 38
COP
Copy File
Source Of4ih0Msg.SourceData[Of4ih0Msg.SourceDataIndex]
Dest Of4ih0Output.MsgRequestBuffer[0]
Length 38
ADD
Add
Source A Of4ih0Msg.SourceBytesSent
3
Source B 38
Dest Of4ih0Msg.SourceBytesSent
3
MOV
Move
Source 38
Dest Of4ih0Msg.SourceTempIndex
4
MOV
Move
Source Of4ih0Msg.SourceBytesSent
3
Dest Of4ih0Msg.SourceDataIndex
3
The following rung determines whether an odd number of bytes have been coppied. If so, the last byte in the last word is padded with a zero.
6
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 1
AND
Bitwise AND
Source A Of4ih0Msg.SourceTempIndex
4
Source B 1
Dest Temp1
256
EQU
Equal
Source A Temp1
256
Source B 1
MOV
Move
Source 0
Dest Of4ih0Output.MsgRequestBuffer[Of4ih0Msg.SourceTempIndex]
1

Figure 6-8d.

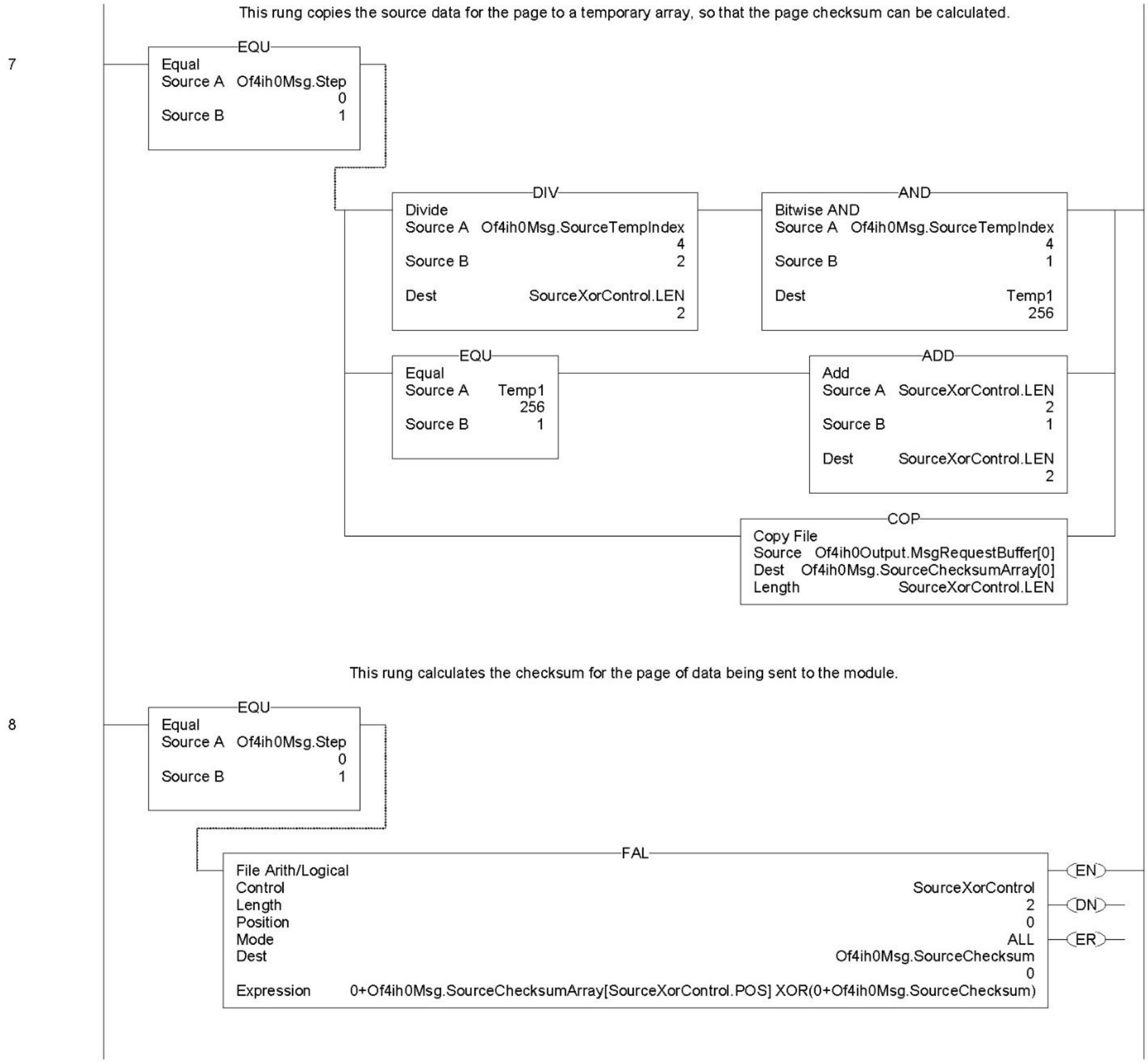

This rung copies the source data for the page to a temporary array, so that the page checksum can be calculated.
7
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 1
DIV
Divide
Source A Of4ih0Msg.SourceTempIndex
4
Source B 2
Dest SourceXorControl.LEN
2
AND
Bitwise AND
Source A Of4ih0Msg.SourceTempIndex
4
Source B 1
Dest Temp1
256
EQU
Equal
Source A Temp1
256
Source B 1
ADD
Add
Source A SourceXorControl.LEN
2
Source B 1
Dest SourceXorControl.LEN
2
COP
Copy File
Source Of4ih0Output.MsgRequestBuffer[0]
Dest Of4ih0Msg.SourceChecksumArray[0]
Length SourceXorControl.LEN
This rung calculates the checksum for the page of data being sent to the module.
8
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 1
FAL
File Arith/Logical
Control SourceXorControl
Length 2
Position 0
Mode ALL
Dest Of4ih0Msg.SourceChecksum
0
Expression 0+Of4ih0Msg.SourceChecksumArray[SourceXorControl.POS] XOR(0+Of4ih0Msg.SourceChecksum)
EN
DN
ER

Figure 6-8e.

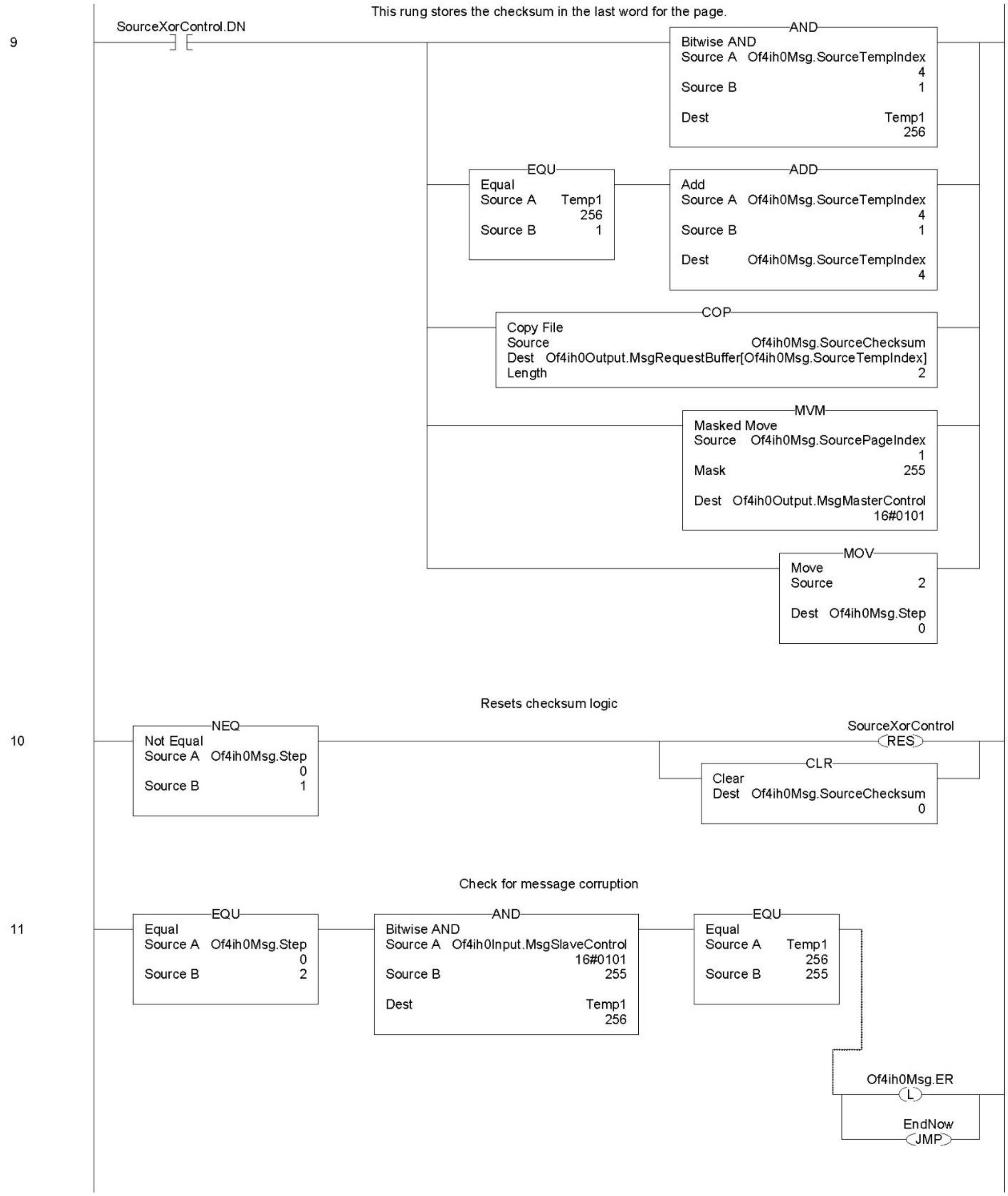

This rung stores the checksum in the last word for the page.
9
SourceXorControl.DN
AND
Bitwise AND
Source A Of4ih0Msg.SourceTempIndex
4
Source B 1
Dest Temp1
256
EQU
Equal
Source A Temp1
256
Source B 1
ADD
Add
Source A Of4ih0Msg.SourceTempIndex
4
Source B 1
Dest Of4ih0Msg.SourceTempIndex
4
COP
Copy File
Source Of4ih0Msg.SourceChecksum
Dest Of4ih0Output.MsgRequestBuffer[Of4ih0Msg.SourceTempIndex]
Length 2
MVM
Masked Move
Source Of4ih0Msg.SourcePageIndex
1
Mask 255
Dest Of4ih0Output.MsgMasterControl
16#0101
MOV
Move
Source 2
Dest Of4ih0Msg.Step
0
Resets checksum logic
10
NEQ
Not Equal
Source A Of4ih0Msg.Step
0
Source B 1
SourceXorControl
RES
CLR
Clear
Dest Of4ih0Msg.SourceChecksum
0
Check for message corruption
11
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 2
AND
Bitwise AND
Source A Of4ih0Input.MsgSlaveControl
16#0101
Source B 255
Dest Temp1
256
EQU
Equal
Source A Temp1
256
Source B 255
Of4ih0Msg.ER
L
EndNow
JMP

Figure 6-8f.

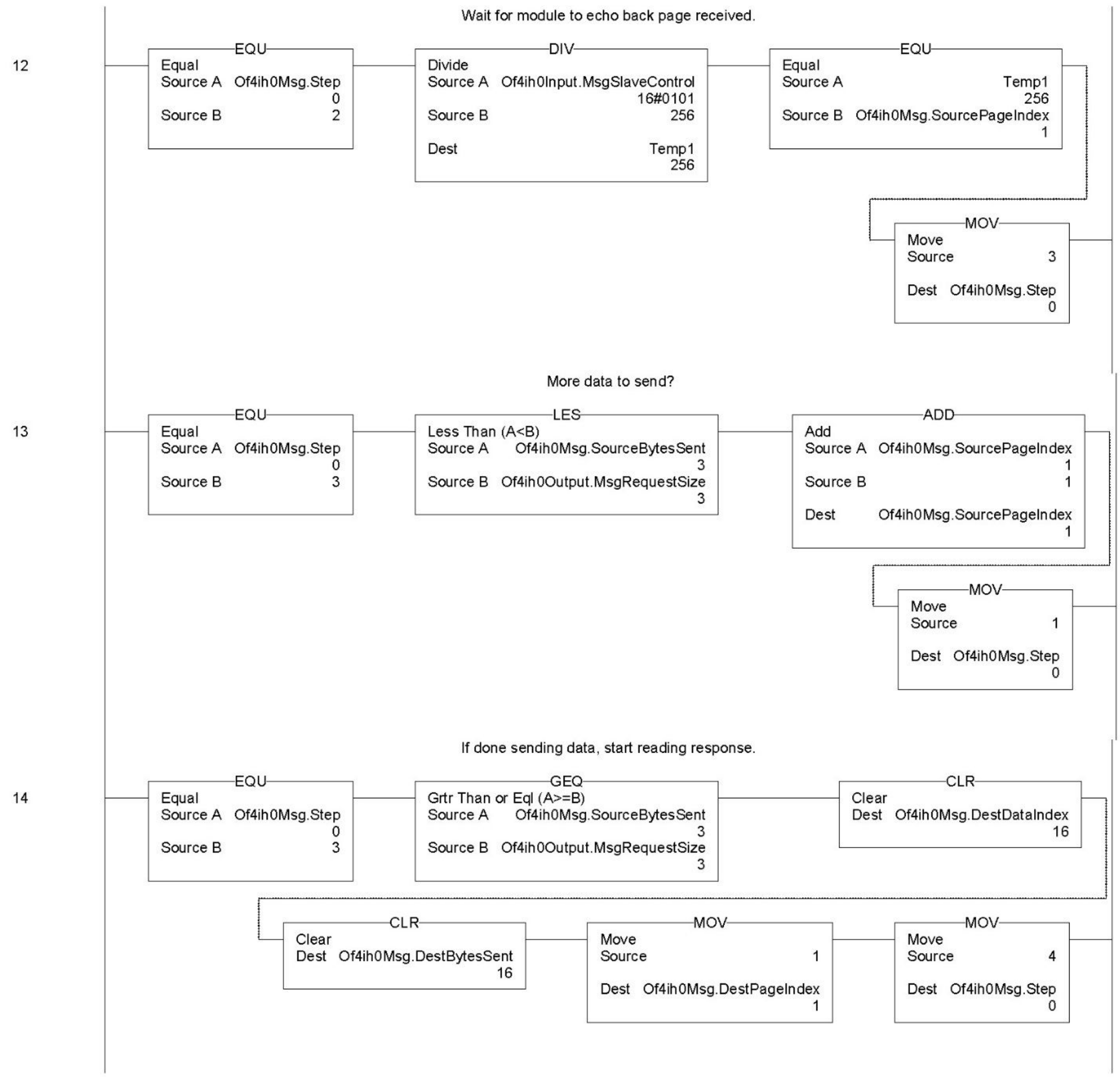

Wait for module to echo back page received.
12
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 2
DIV
Divide
Source A Of4ih0Input.MsgSlaveControl
16#0101
Source B 256
Dest Temp1
256
EQU
Equal
Source A Temp1
256
Source B Of4ih0Msg.SourcePageIndex
1
MOV
Move
Source 3
Dest Of4ih0Msg.Step
0
More data to send?
13
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 3
LES
Less Than (A<B)
Source A Of4ih0Msg.SourceBytesSent
3
Source B Of4ih0Output.MsgRequestSize
3
ADD
Add
Source A Of4ih0Msg.SourcePageIndex
1
Source B 1
Dest Of4ih0Msg.SourcePageIndex
1
MOV
Move
Source 1
Dest Of4ih0Msg.Step
0
If done sending data, start reading response.
14
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 3
GEQ
Grtr Than or Eql (A>=B)
Source A Of4ih0Msg.SourceBytesSent
3
Source B Of4ih0Output.MsgRequestSize
3
CLR
Clear
Dest Of4ih0Msg.DestDataIndex
16
CLR
Clear
Dest Of4ih0Msg.DestBytesSent
16
MOV
Move
Source 1
Dest Of4ih0Msg.DestPageIndex
1
MOV
Move
Source 4
Dest Of4ih0Msg.Step
0

Figure 6-8g.

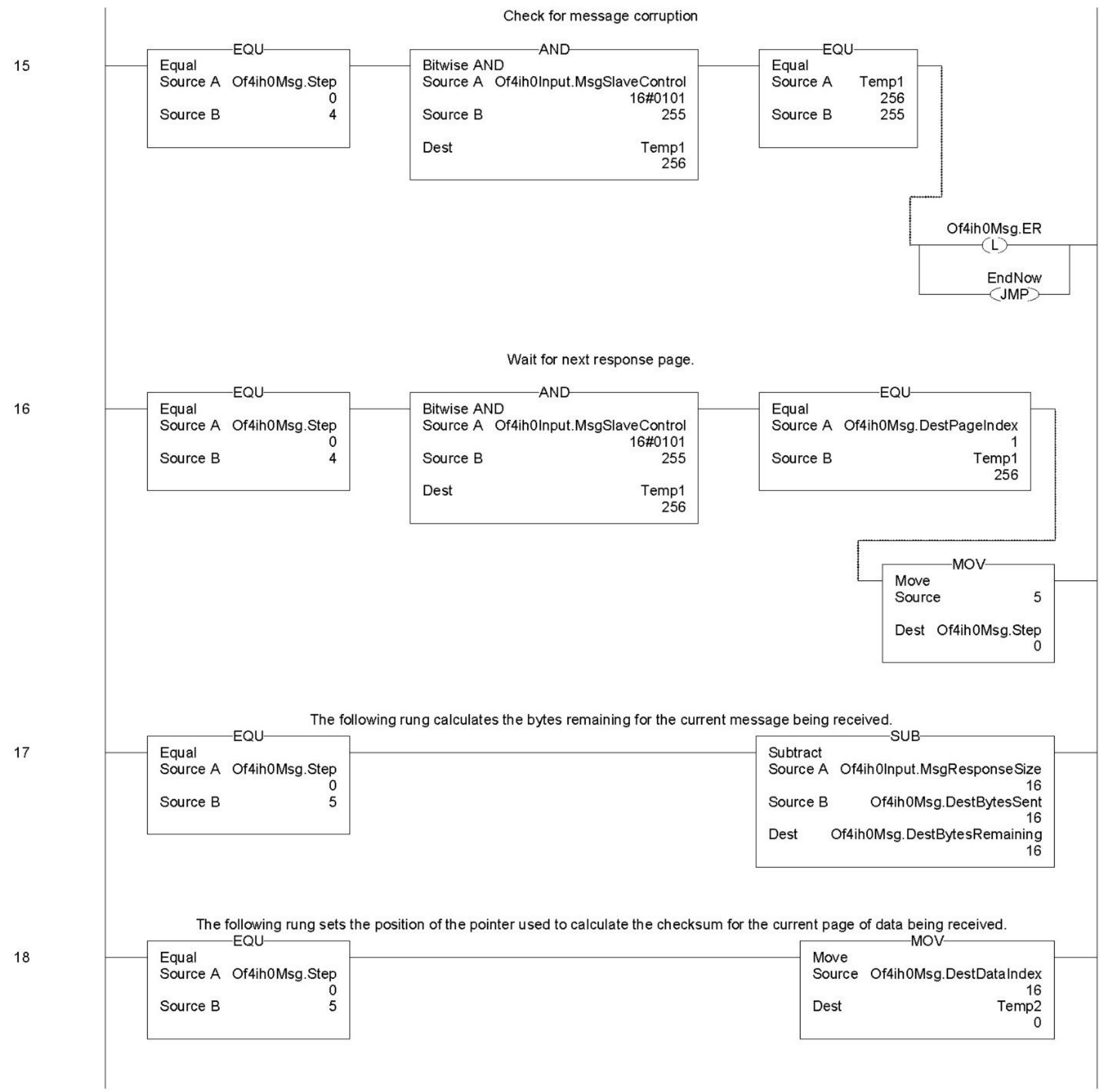

Check for message corruption
15
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 4
AND
Bitwise AND
Source A Of4ih0Input.MsgSlaveControl
16#0101
Source B 255
Dest Temp1
256
EQU
Equal
Source A Temp1
256
Source B 255
Of4ih0Msg.ER
L
EndNow
JMP
Wait for next response page.
16
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 4
AND
Bitwise AND
Source A Of4ih0Input.MsgSlaveControl
16#0101
Source B 255
Dest Temp1
256
EQU
Equal
Source A Of4ih0Msg.DestPageIndex
1
Source B Temp1
256
MOV
Move
Source 5
Dest Of4ih0Msg.Step
0
The following rung calculates the bytes remaining for the current message being received.
17
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 5
SUB
Subtract
Source A Of4ih0Input.MsgResponseSize
16
Source B Of4ih0Msg.DestBytesSent
16
Dest Of4ih0Msg.DestBytesRemaining
16
The following rung sets the position of the pointer used to calculate the checksum for the current page of data being received.
18
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 5
MOV
Move
Source Of4ih0Msg.DestDataIndex
16
Dest Temp2
0

Figure 6-8h.

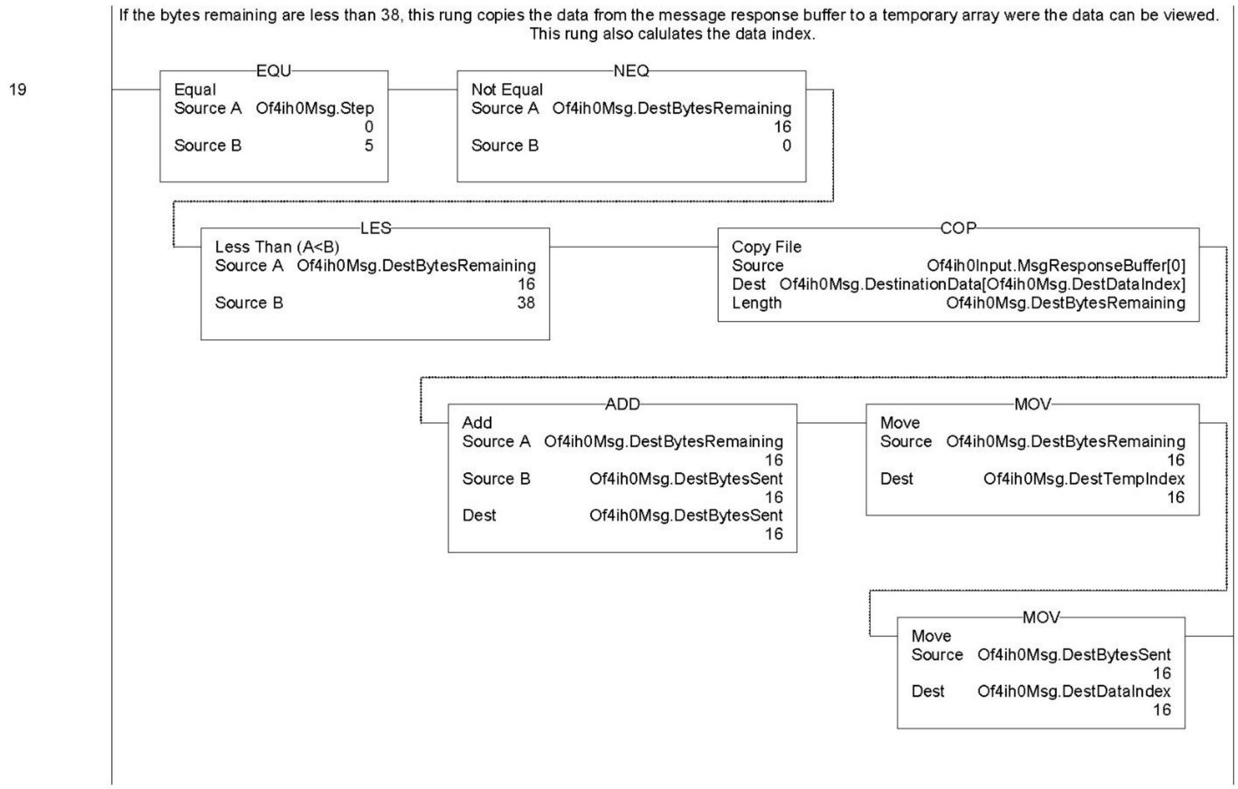

If the bytes remaining are less than 38, this rung copies the data from the message response buffer to a temporary array were the data can be viewed.
This rung also calulates the data index.
19
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 5
NEQ
Not Equal
Source A Of4ih0Msg.DestBytesRemaining
16
Source B 0
LES
Less Than (A<B)
Source A Of4ih0Msg.DestBytesRemaining
16
Source B 38
COP
Copy File
Source Of4ih0Input.MsgResponseBuffer[0]
Dest Of4ih0Msg.DestinationData[Of4ih0Msg.DestDataIndex]
Length Of4ih0Msg.DestBytesRemaining
ADD
Add
Source A Of4ih0Msg.DestBytesRemaining
16
Source B Of4ih0Msg.DestBytesSent
16
Dest Of4ih0Msg.DestBytesSent
16
MOV
Move
Source Of4ih0Msg.DestBytesRemaining
16
Dest Of4ih0Msg.DestTempIndex
16
MOV
Move
Source Of4ih0Msg.DestBytesSent
16
Dest Of4ih0Msg.DestDataIndex
16

Figure 6-8i.

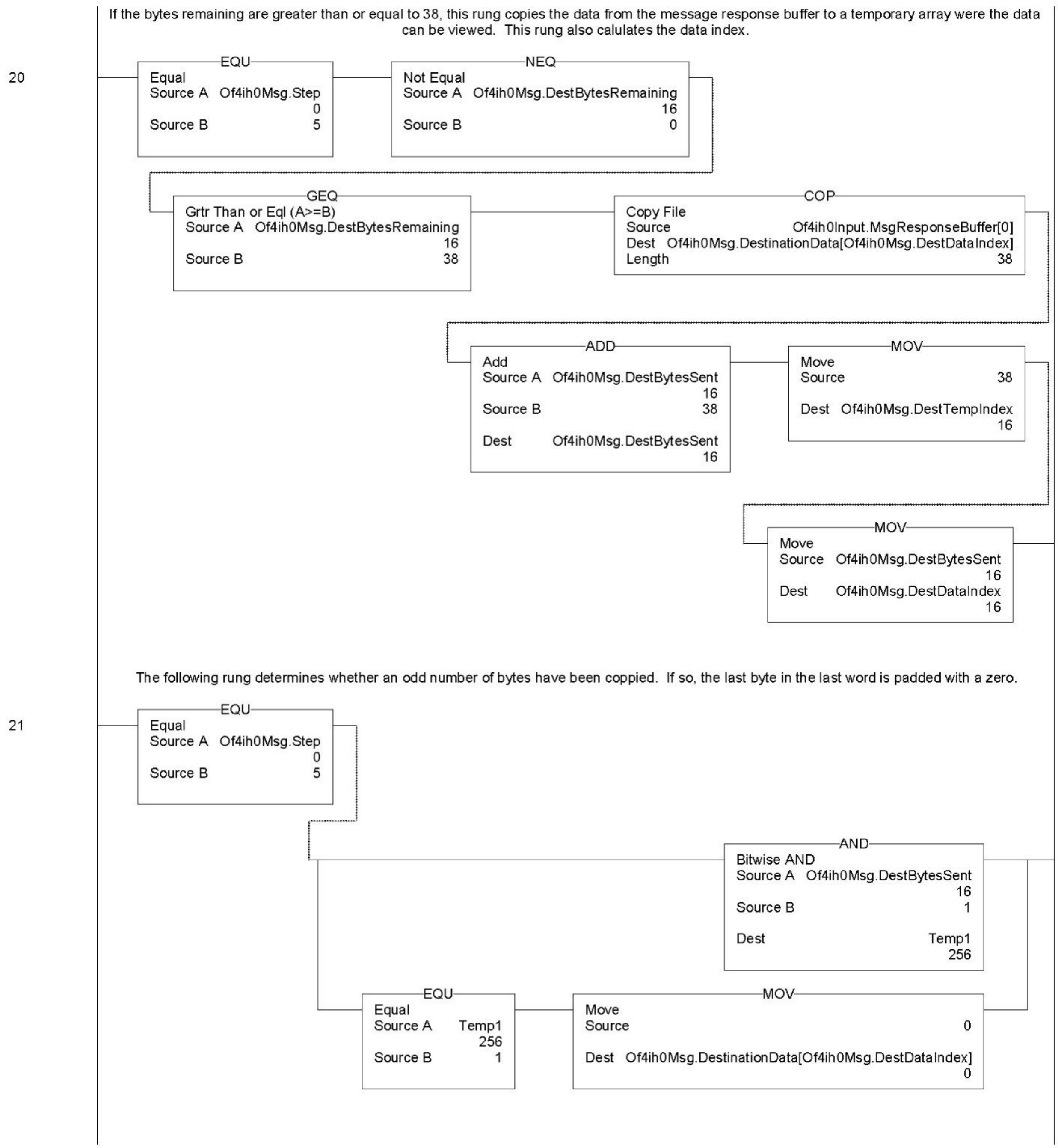

If the bytes remaining are greater than or equal to 38, this rung copies the data from the message response buffer to a temporary array were the data can be viewed. This rung also calulates the data index.
20
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 5
NEQ
Not Equal
Source A Of4ih0Msg.DestBytesRemaining
16
Source B 0
GEQ
Grtr Than or Eql (A>=B)
Source A Of4ih0Msg.DestBytesRemaining
16
Source B 38
COP
Copy File
Source Of4ih0Input.MsgResponseBuffer[0]
Dest Of4ih0Msg.DestinationData[Of4ih0Msg.DestDataIndex]
Length 38
ADD
Add
Source A Of4ih0Msg.DestBytesSent
16
Source B 38
Dest Of4ih0Msg.DestBytesSent
16
MOV
Move
Source 38
Dest Of4ih0Msg.DestTempIndex
16
MOV
Move
Source Of4ih0Msg.DestBytesSent
16
Dest Of4ih0Msg.DestDataIndex
16
The following rung determines whether an odd number of bytes have been coppied. If so, the last byte in the last word is padded with a zero.
21
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 5
AND
Bitwise AND
Source A Of4ih0Msg.DestBytesSent
16
Source B 1
Dest Temp1
256
EQU
Equal
Source A Temp1
256
Source B 1
MOV
Move
Source 0
Dest Of4ih0Msg.DestinationData[Of4ih0Msg.DestDataIndex]
0

Figure 6-8j.

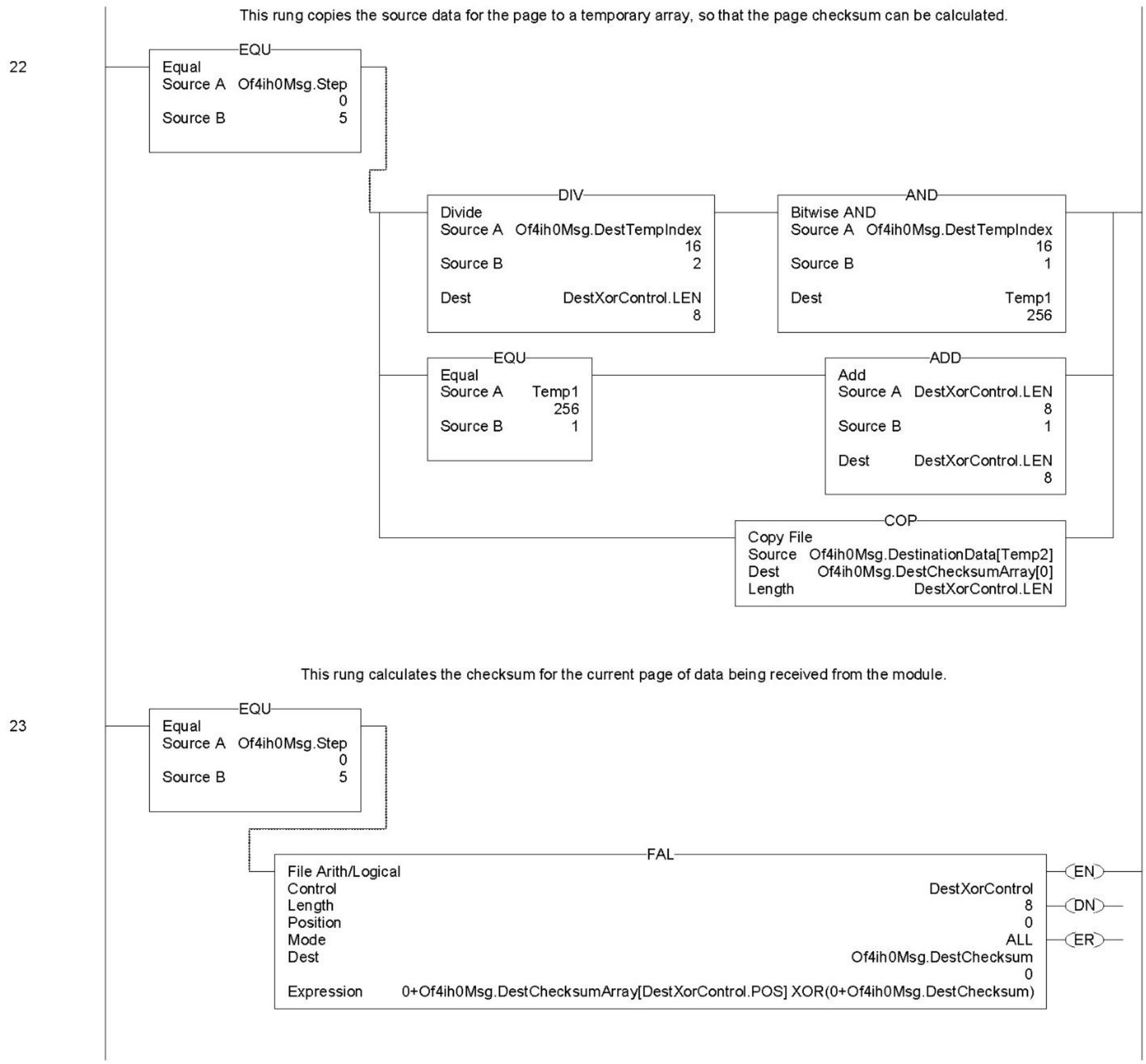

This rung copies the source data for the page to a temporary array, so that the page checksum can be calculated.
22
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 5
DIV
Divide
Source A Of4ih0Msg.DestTempIndex
16
Source B 2
Dest DestXorControl.LEN
8
AND
Bitwise AND
Source A Of4ih0Msg.DestTempIndex
16
Source B 1
Dest Temp1
256
EQU
Equal
Source A Temp1
256
Source B 1
ADD
Add
Source A DestXorControl.LEN
8
Source B 1
Dest DestXorControl.LEN
8
COP
Copy File
Source Of4ih0Msg.DestinationData[Temp2]
Dest Of4ih0Msg.DestChecksumArray[0]
Length DestXorControl.LEN
This rung calculates the checksum for the current page of data being received from the module.
23
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 5
FAL
File Arith/Logical
Control DestXorControl
Length 8
Position 0
Mode ALL
Dest Of4ih0Msg.DestChecksum
0
Expression 0+Of4ih0Msg.DestChecksumArray[DestXorControl.POS] XOR(0+Of4ih0Msg.DestChecksum)
EN
DN
ER

Figure 6-8k.

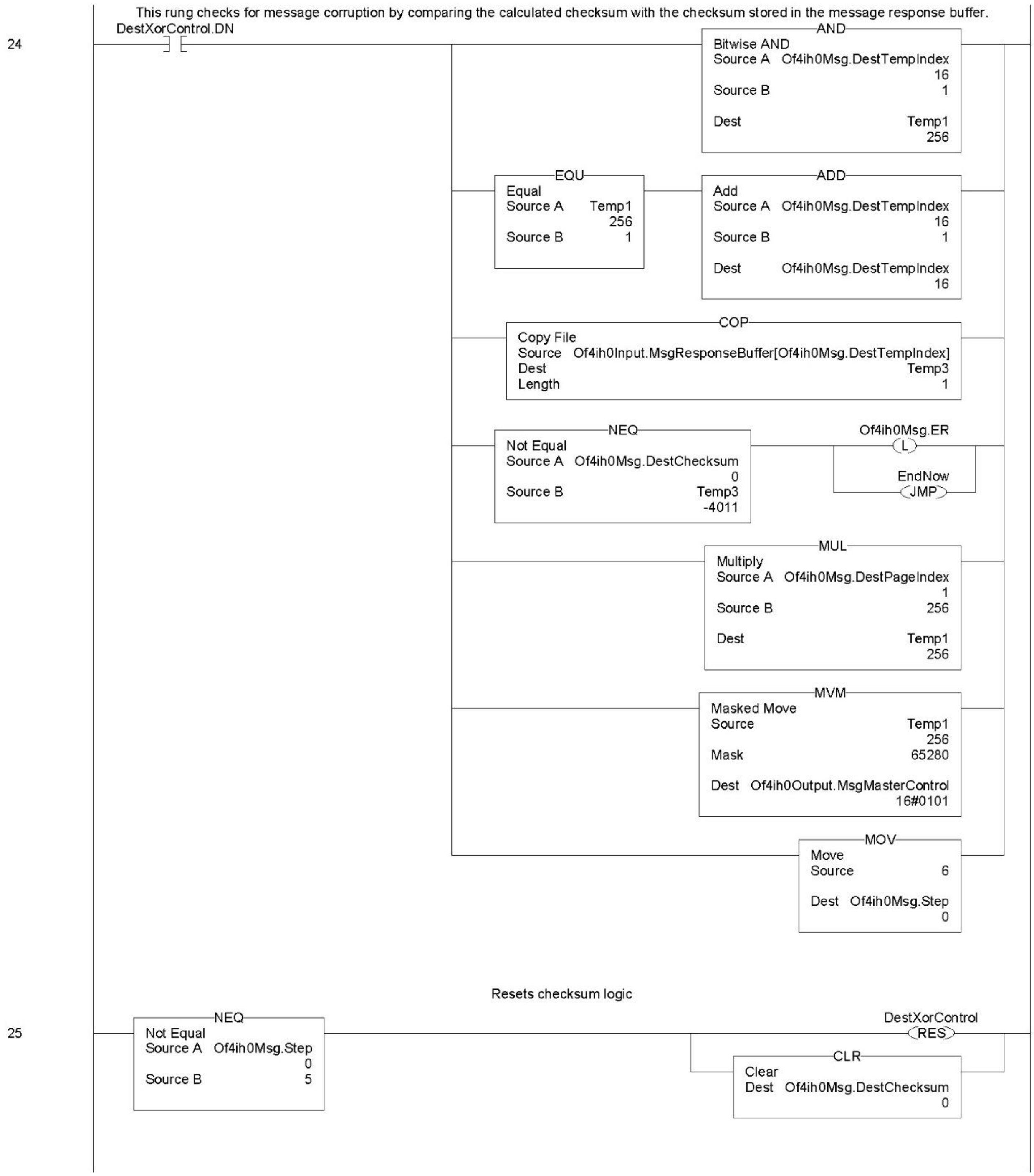
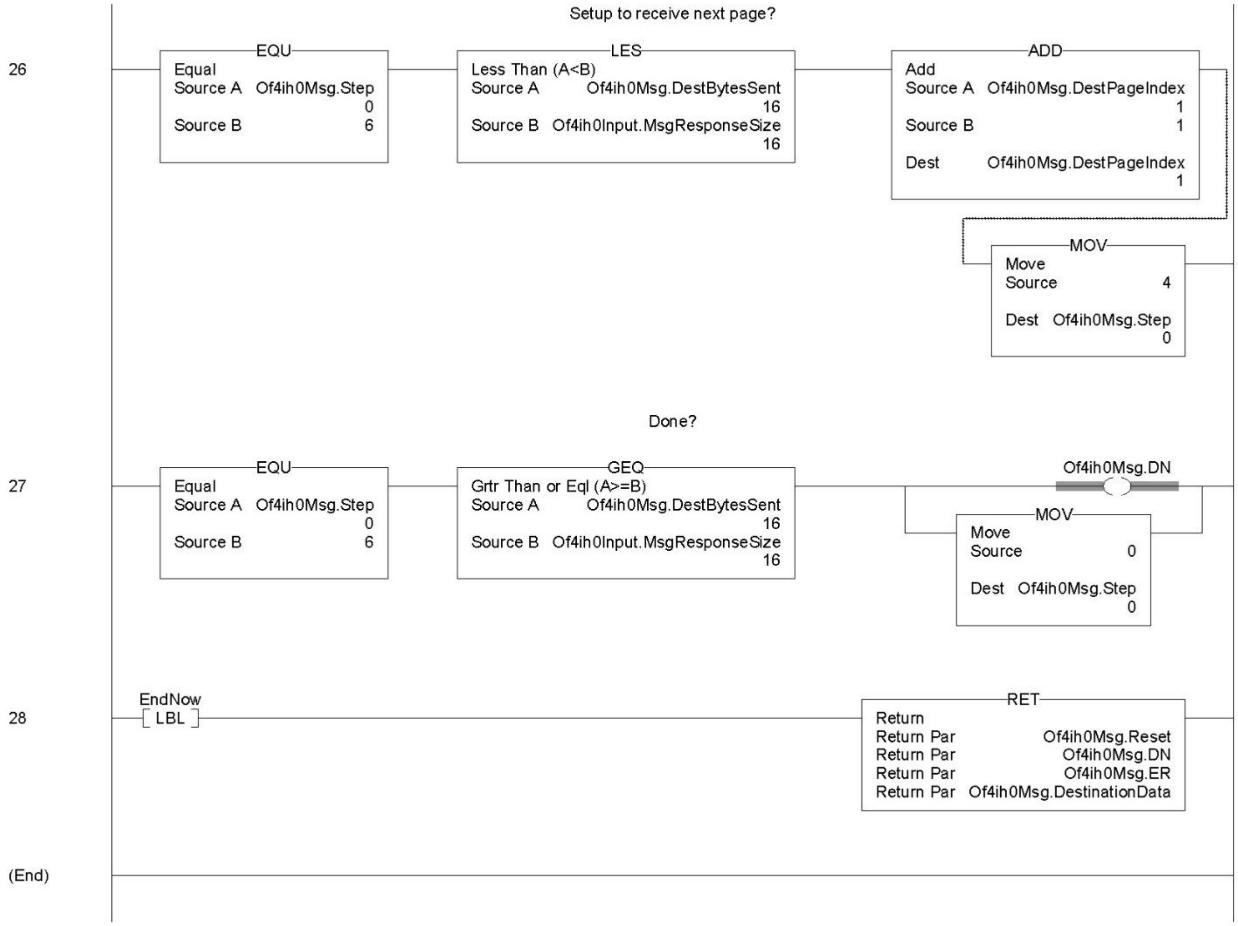

This rung checks for message corruption by comparing the calculated checksum with the checksum stored in the message response buffer.
24
DestXorControl.DN
AND
Bitwise AND
Source A Of4ih0Msg.DestTempIndex
16
Source B 1
Dest Temp1
256
EQU
Equal
Source A Temp1
256
Source B 1
ADD
Add
Source A Of4ih0Msg.DestTempIndex
16
Source B 1
Dest Of4ih0Msg.DestTempIndex
16
COP
Copy File
Source Of4ih0Input.MsgResponseBuffer[Of4ih0Msg.DestTempIndex]
Dest Temp3
Length 1
NEQ
Not Equal
Source A Of4ih0Msg.DestChecksum
0
Source B Temp3
-4011
Of4ih0Msg.ER
L
EndNow
JMP
MUL
Multiply
Source A Of4ih0Msg.DestPageIndex
1
Source B 256
Dest Temp1
256
MVM
Masked Move
Source Temp1
256
Mask 65280
Dest Of4ih0Output.MsgMasterControl
16#0101
MOV
Move
Source 6
Dest Of4ih0Msg.Step
0
Resets checksum logic
25
NEQ
Not Equal
Source A Of4ih0Msg.Step
0
Source B 5
DestXorControl
RES
CLR
Clear
Dest Of4ih0Msg.DestChecksum
0

Figure 6-8l.

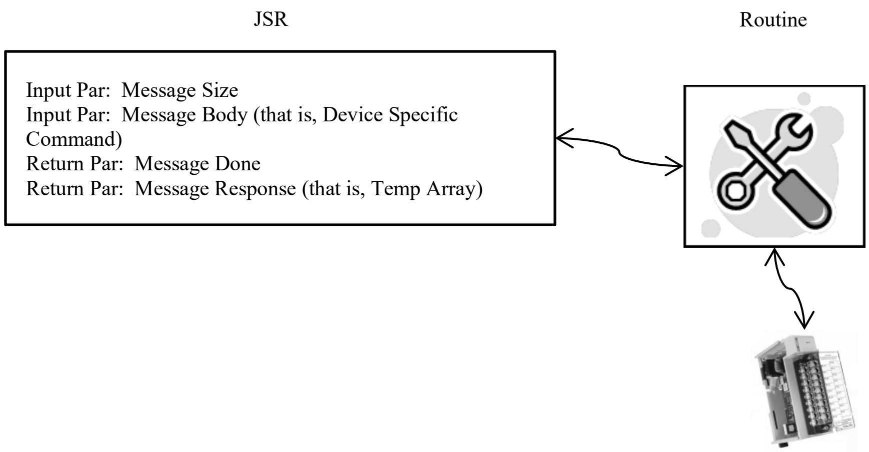

Setup to receive next page?
26
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 6
LES
Less Than (A<B)
Source A Of4ih0Msg.DestBytesSent
16
Source B Of4ih0Input.MsgResponseSize
16
ADD
Add
Source A Of4ih0Msg.DestPageIndex
1
Source B 1
Dest Of4ih0Msg.DestPageIndex
1
MOV
Move
Source 4
Dest Of4ih0Msg.Step
0
Done?
27
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 6
GEQ
Grtr Than or Eql (A>=B)
Source A Of4ih0Msg.DestBytesSent
16
Source B Of4ih0Input.MsgResponseSize
16
Of4ih0Msg.DN
MOV
Move
Source 0
Dest Of4ih0Msg.Step
0
28
EndNow
LBL
RET
Return
Return Par Of4ih0Msg.Reset
Return Par Of4ih0Msg.DN
Return Par Of4ih0Msg.ER
Return Par Of4ih0Msg.DestinationData
(End)

Section 6.4 Module Specific Commands

The HART output module uses module-specific commands. Module-specific commands include the HART pass-through, HART suspend and resume, and get HART device information. The commands are passed to the module using the input and output image. Since some messages can be 257 bytes long, the data is transported to, and from, the module 40 bytes at a time using the paging scheme described in the previous section.

The module specific command and accompanying data is passed to the routine in Figure 6-9 using a JSR instruction with parameters. When the routine is executed it will send the message to the module. The response data, if any, is also converted by this routine and stored in a temporary array where it can be used within the ladder program. See figure below.

Figure 6-9. Message Flow



The tables on the following pages show the format for each module-specific command.

6.4.1 Get HART Device Information

The *Get HART Device Information* command is used to gather the device-specific information for the connected HART device. The data that is retrieved can be seen in the following tables The information that is gathered by this command is similar to the information gathered from the auto-acquisition process. The key difference is that the *Get HART Device Information* command pulls the data that has been stored in the module RAM and not directly from the field device.

Table 6-9a. Get HART Device Information Command

HART Get Device Information – command message packet structure Get currently cached Device Information for a given channel:		
Field	Value	Definition
HART Channel Number	0×00 – 0×03 (1 byte)	Module input channel number for HART command

HART Get Device Information – command message packet structure Get currently cached Device Information for a given channel:		
Field	Value	Definition
Command Number	0×03 (1 byte)	The command number to obtain HART device information

Table 6-9b. Response If Device Information Is Not Available

HART Get Device Information - reply packet structure		
Field	Value	Definition
HART Channel Number	0×00 – 0×03 (1 byte)	Module input channel number for HART command
Status	(1 byte) 34 = DR_RUNNING 35 = DR_DEAD (bad request)	Command status
Count	(1 byte)	Set to 1
Handle	0	Fill byte of zero to keep command response common among all replies.

Table 6-10. Response When Device Information Is Available

HART Get Device Information - reply packet structure		
Field	Value	Definition
HART Channel Number	0×00 – 0×03 (1 byte)	Module input channel number for HART command
Status	00 = SUCCESS	Command status
Count	(1 byte)	Number of data bytes to following.
HART ManufacturerIDCode	(1 byte)	CMD#0, Byte 1
HARTDeviceTypeCode	(1 byte)	CMD#0, Byte 2
HARTPreamble	(1 byte)	CMD#0, Byte 3
HARTUnivCmdCode	(1 byte)	CMD#0, Byte 4
HARTTransSpecRev	(1 byte)	CMD#0, Byte 5
HARTSoftwareRevision	(1 byte)	CMD#0, Byte 6
HARTHardwareRevision	(1 byte)	CMD#0, Byte 7
HARTFlags	(1 byte)	CMD#0, Byte 8
Pad for 32-bit alignment	(1 byte)	

HART Get Device Information - reply packet structure		
Field	Value	Definition
HARTDeviceIDNumber	(3 bytes) Device ID number	CMD#0, Bytes 9-11
Pad for 32-bit alignment	(1 byte)	
HARTTag	(8 bytes unpacked ASCII)	CMD#13, Bytes 0-5
HARTDescriptor	(16 bytes unpacked ASCII)	CMD#13, Bytes 6-17
HARTDate	(3 bytes)	CMD#13, Bytes 18-20
Pad for 32-bit alignment	(1 byte)	
HARTFinalAssemblyNumber	(3 bytes)	CMD#16, Bytes 0-2
Pad for 32-bit alignment	(1 byte)	
HARTMessage	(32 bytes unpacked ASCII)	CMD#12, Bytes 0-23
HARTPVCode	(1 byte)	CMD#50, Bytes 0, 0×ff if not supported
HARTSVCode	(1 byte)	CMD#50, Bytes 1, 0×ff if not supported
HARTTVCode	(1 byte)	CMD#50, Bytes 2, 0×ff if not supported
HARTQVCode	(1 byte)	CMD#50, Bytes 3, 0×ff if not supported
HARTPVUnits	(1 byte)	CMD#3, Byte 4
HARTSVUnits	(1 byte)	CMD#3, Byte 9, 0 if not present
HARTTVUnits	(1 byte)	CMD#3, Byte 14, 0 if not present
HARTQVUnits	(1 byte)	CMD#3, Byte 19, 0 if not present
HARTSlot0Units	(1 byte)	CMD#33, Byte 1, 0 if not present Output module use only.
HARTSlot1Units	(1 byte)	CMD#33, Byte 7, 0 if not present Output module use only.
HARTSlot2Units	(1 byte)	CMD#33, Byte 13, 0 if not present

HART Get Device Information - reply packet structure		
Field	Value	Definition
		Output module use only.
HARTSlot3Units	(1 byte)	CMD#33, Byte 19, 0 if not present Output module use only.
HARTPVLowerRange	(4 bytes – Floating Point Value)	CMD#15, Bytes 3-6
HARTPVUpperRange	(4 bytes – Floating Point Value)	CMD#15, Bytes 7-10
Pad for 32-bit alignment	(3 bytes)	

The command status, the second byte in the reply packet for the module specific command, can return three different responses, SUCCESS, RUNNING, and DEAD. These responses echo the state of the module at the time the command is sent. The conditions for each response are as follows:

SUCCESS will be sent back when all the following conditions are met:

- Command and HART Channel number are both valid.
- HART channel device information is available.

RUNNING will be sent back when all the following conditions are met:

- Command and HART Channel number are both valid.
- HART channel is enabled, and communication has been established, meaning at least the device addressing information is available.
- HART channel is already in the state of gathering device information. Reply will be sent back without additional events triggered.

DEAD will be sent back if any of the following conditions is true:

- Command or HART Channel number is invalid.
- HART channel is not enabled.
- HART communication has not been established, meaning that the 5-byte unique address has not been determined yet.
- All other conditioned not generating RUNNING or SUCCESS.

6.4.2 HART Pass-Through Command

The HART Pass-Through Command can be used to send any HART command including universal, common practice or device-specific, directly to a field device. The module in this case could be considered a HART bridge. There can be two (2) instances of a HART pass-through message being serviced, meaning the pass-through message queue is 2 deep. The HART pass-through response will be queued the moment the command is received, if the queue spaces are not already in use, and be dispatched after at least a full scan is done. In other words, after servicing a pass-through, the HART module will make sure all enabled HART channels have updated variable values before another pass-through is placed into service.

All HART pass-through commands require a series of messages to be exchanged. First, a pass-through command request must be sent to the HART module to initiate the pass-through command. The HART module will respond to the command request with a command request reply that includes a handle that can be used to obtain the pass-through message response. Once the handle is received, the user may issue a Get Command Query to obtain the status of the pass-through command and the pass-through command response data, if it is available.

There is a handle timeout associated with the final reply message. After the HART module obtains the requested information from the HART device, it will start a handle timeout timer. Refer to Chapter 5 for information regarding how to set the handle timeout. The reply message will be kept persistent during the handle timeout period. When the handle timeout timer expires the reply message will be discarded, and another pass-through message will be serviced without being rejected. The user-defined handle timeout is in the range of 1 to 255 seconds.

NOTE	If the HART message being sent or received using the pass-through command contains floating point values, the order of the bytes must be reversed.

Depending on the HART command, the data contained within the HART message may include floating point numbers or double integers. If a floating point or double integer is contained within the HART message, the user must be aware that the order of the bytes that make up the float or double will need to be reversed. The reason for this is related to how the bytes are stored in the ControlLogix processor.

The ControlLogix processor stores the bytes in memory in a format referred to as "little-endian". Little-endian is an order in which the "little end" (least significant value in the sequence) is stored first (at the lowest storage address). However, HART devices transmit the byte data in the reverse order or as you may have guessed "big-endian". Refer to Chapter 7 for a ladder sample demonstrating the process of swapping the order of the bytes.

Table 6-9. HART Pass-Through Request Command

HART pass-through command request – command message packet structure		
Field	Value	Definition
HART Channel Number	0×00 – 0×03 (1 byte)	Module input channel number for HART command
Command Number	0×01 (1 byte)	The command number to issue a HART pass-through command.

HART pass-through command request – command message packet structure		
Field	Value	Definition
HART Command	N bytes N = Length of message – 2 Contents are as follows: Start or Delimiter (1 byte): 0×82 Long form Address (5 bytes) HART Command number (1 byte) Request Data Count (1 byte) Data (“Request Data Count” bytes) Checksum (XOR of all bytes from delimiter on. Delimiter is included.)	The actual HART command PDU

Table 6-10. HART Pass-Through Request Reply

HART pass-through command request – reply packet structure		
Field	Value	Definition
HART Channel Number	0×00 – 0×03 (1 byte)	Module input channel number for HART command
Status	(1 byte) 32 = Busy (Queue is already full). 33 = DR_INITIATE 35 = DR_DEAD (bad request)	Command status
Count	(1 byte)	Set to 1
Handle	(1 byte) 0 (bad when status is DR_DEAD) 1-255 (good)	The handle for command complete query

The command status, the second byte in the reply packet for this module-specific command, can return two different responses, INITIATE, and DEAD. These responses echo the state of the module at the time the command is sent. The conditions for each response are as follows:

INITIATE will be sent back under the following conditions:

- Command and HART Channel number are both valid.
- HART channel is enabled, and communication has been established, meaning at least the device addressing information is available.
- Handle is available, meaning no pending handle is still active.
- HART channel is doing regular data sampling only. No pending device information gathering is active.
- No pending pass-through handle is active, meaning handle timeout has not occurred yet.
- Device address and delimiter are valid.

- Received CIP word count is large enough for the entire command packet.

DEAD will be sent back if any of the following conditions are true:

- Command or HART Channel number is invalid.
- HART channel is not enabled.
- HART communication has not been established, meaning that the 5-byte unique address has not been determined yet.
- The channel is currently updating device information. Theoretically, pass-through command can be safely accepted after successfully receiving Command 0, but for simplicity, we'll track update of the device information as a whole.
- All other conditioned not generating INITIATE.

After the pass-through response is sent with a valid handle and a response value indicating (33) INITIATE, the user can retrieve the data associated with the handle using the following command message.

Table 6-11. HART Pass-Through Query Command

HART pass-through command complete query - command message packet structure		
Field	Value	Definition
HART Channel Number	0×00 – 0×03 (1 byte)	Module input channel number for HART command
Command Number	0×0C (1 byte)	The command number
Handle	(1 byte) 1-255	The handle from command request reply

If the data associated with the handle is not yet available, or invalid, the following reply message will be returned.

Table 6-12. HART Pass-Through Query Reply NOT SUCCESS

HART pass-through command complete query - reply packet structure		
Field	Value	Definition
Unconnected Message Header		
HART Channel Number	0×00 – 0×07 (1 byte)	Module input channel number for HART command
Status	(1 byte) 34 = DR_RUNNING 35 = DR_DEAD (bad request)	Command status
Count	(2 bytes) (Command Number 0×0C)	Length of Handle + HART Response Data in bytes (if Success)

HART pass-through command complete query - reply packet structure		
Field	Value	Definition
Unconnected Message Header		
Handle	(1 byte)	The handle from command complete query

When data associated with the buffer becomes available, meaning a "success" response, the reply will be formatted as follows:

Table 6-13. HART Pass-Through Query Reply SUCCESS

HART pass-through command complete query - reply packet structure		
Field	Value	Definition
Unconnected Message Header		
HART Channel Number	0×00 – 0×07 (1 byte)	Module input channel number for HART command
Status	(1 byte) 00 = SUCCESS	Command status
Count	(1 byte) (Command Number 0×04) (2 bytes) (Command Number 0×08, 0×0C)	Length of Handle + HART Response Data in bytes (if Success)
Handle	1-255	The handle from command complete query
HART Command Response Data	Size is the entire HART device response size in bytes. The size does not include preambles bytes.	The HART device’s response to the command (if Success)

The command status, the second byte in the reply packet for the module-specific command, can return three different responses, SUCCESS, RUNNING, and DEAD. These responses echo the state of the module at the time the command is sent. The conditions for each response are as follows:

SUCCESS will be sent back under the following conditions:

- Command and HART Channel number are both valid.
- HART channel is enabled.
- Command handle matches currently active handle, and the handle is in the HOLD state.
- After replying with a SUCCESS, the handle will become inactive, thus allowing for next pass-through or host-initiated update of device information.

RUNNING will be sent back under the following conditions:

- Command and HART Channel number are both valid.

- HART channel is enabled.
- Command handle matches currently active handle.
- HART channel is already in the state of handling a pass-through command. Reply will be sent back without additional events triggered.

DEAD will be sent back if any of the following conditions are true:

- Command or HART Channel number is invalid.
- HART channel is not enabled.
- HART communication has not been established, meaning that the 5-byte unique address has not been determined yet.
- All other conditions not generating RUNNING or SUCCESS. Examples: invalid handle, handle timed out, channel under device information gathering, and etc.

The following ladder demonstrates how to perform the pass-through request and query process.

Figure 6-10a. Pass-Through Ladder

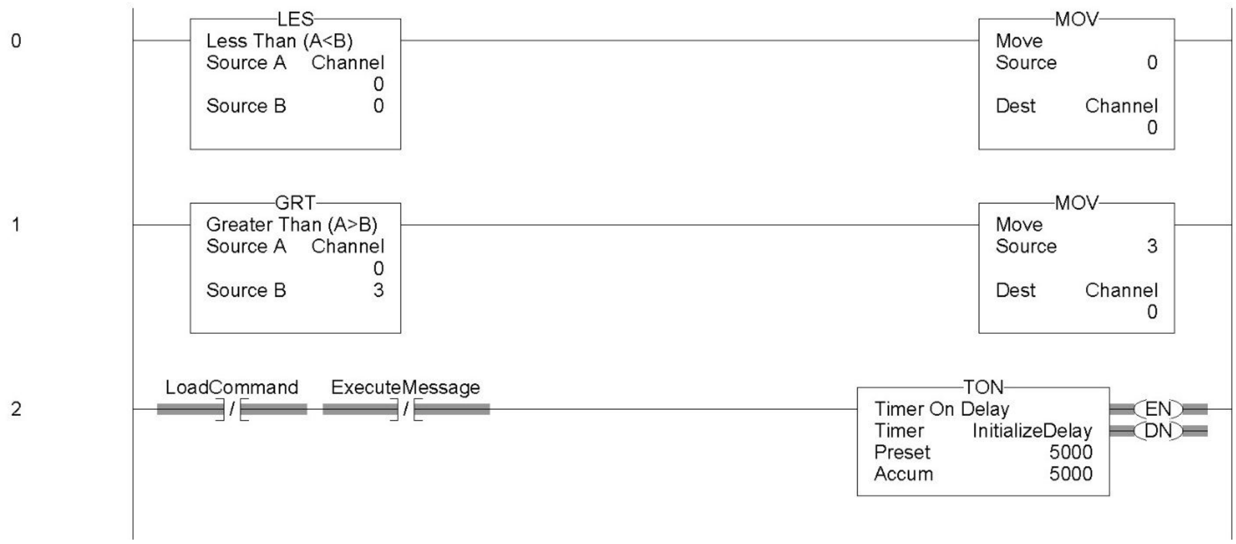
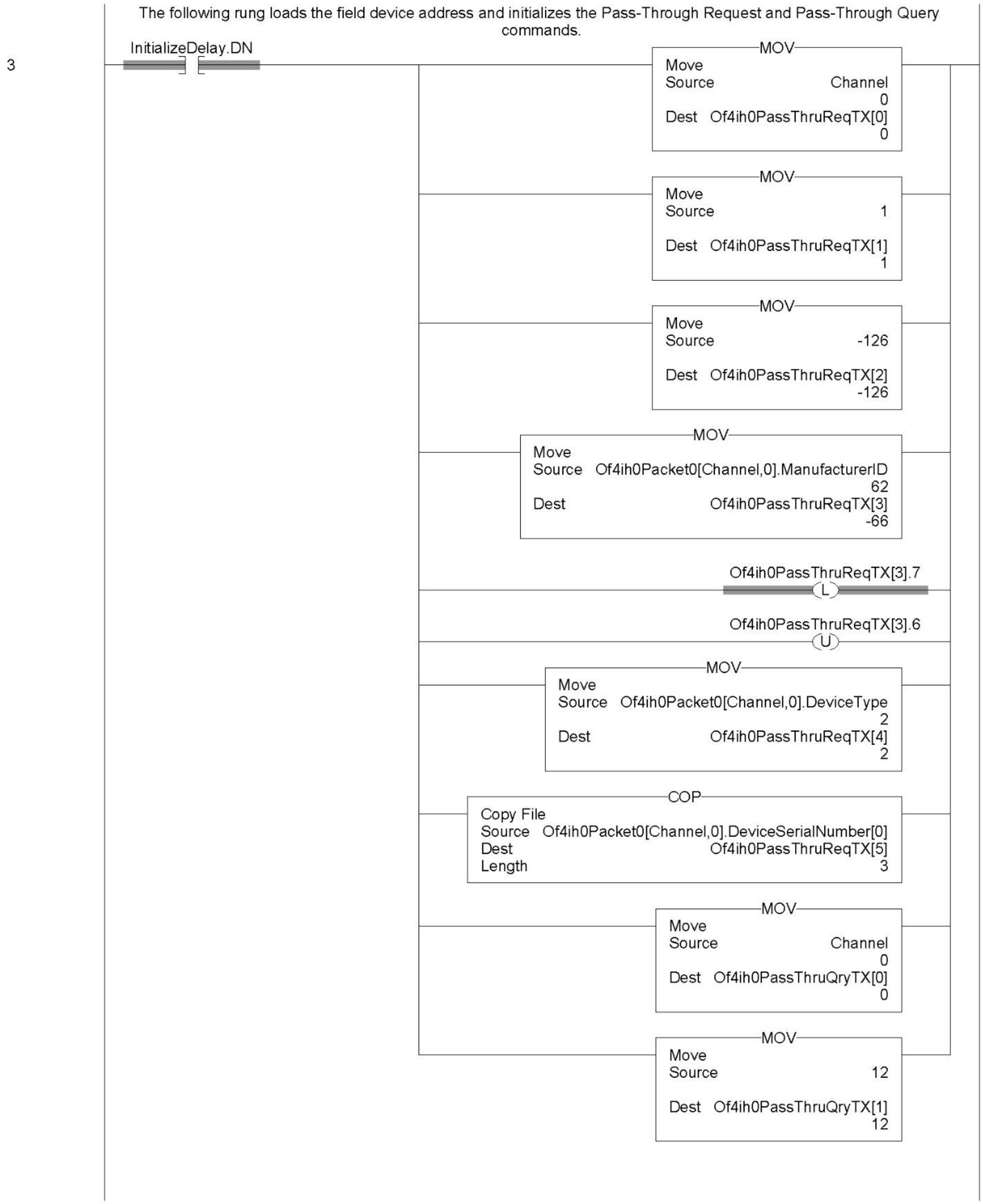


Figure 6-10b.



The following rung loads the field device address and initializes the Pass-Through Request and Pass-Through Query commands.
3
InitializeDelay.DN
MOV
Move
Source Channel
0
Dest Of4ih0PassThruReqTX[0]
0
MOV
Move
Source 1
Dest Of4ih0PassThruReqTX[1]
1
MOV
Move
Source -126
Dest Of4ih0PassThruReqTX[2]
-126
MOV
Move
Source Of4ih0Packet0[Channel,0].ManufacturerID
62
Dest Of4ih0PassThruReqTX[3]
-66
Of4ih0PassThruReqTX[3].7
L
Of4ih0PassThruReqTX[3].6
U
MOV
Move
Source Of4ih0Packet0[Channel,0].DeviceType
2
Dest Of4ih0PassThruReqTX[4]
2
COP
Copy File
Source Of4ih0Packet0[Channel,0].DeviceSerialNumber[0]
Dest Of4ih0PassThruReqTX[5]
Length 3
MOV
Move
Source Channel
0
Dest Of4ih0PassThruQryTX[0]
0
MOV
Move
Source 12
Dest Of4ih0PassThruQryTX[1]
12

Figure 6-10c.

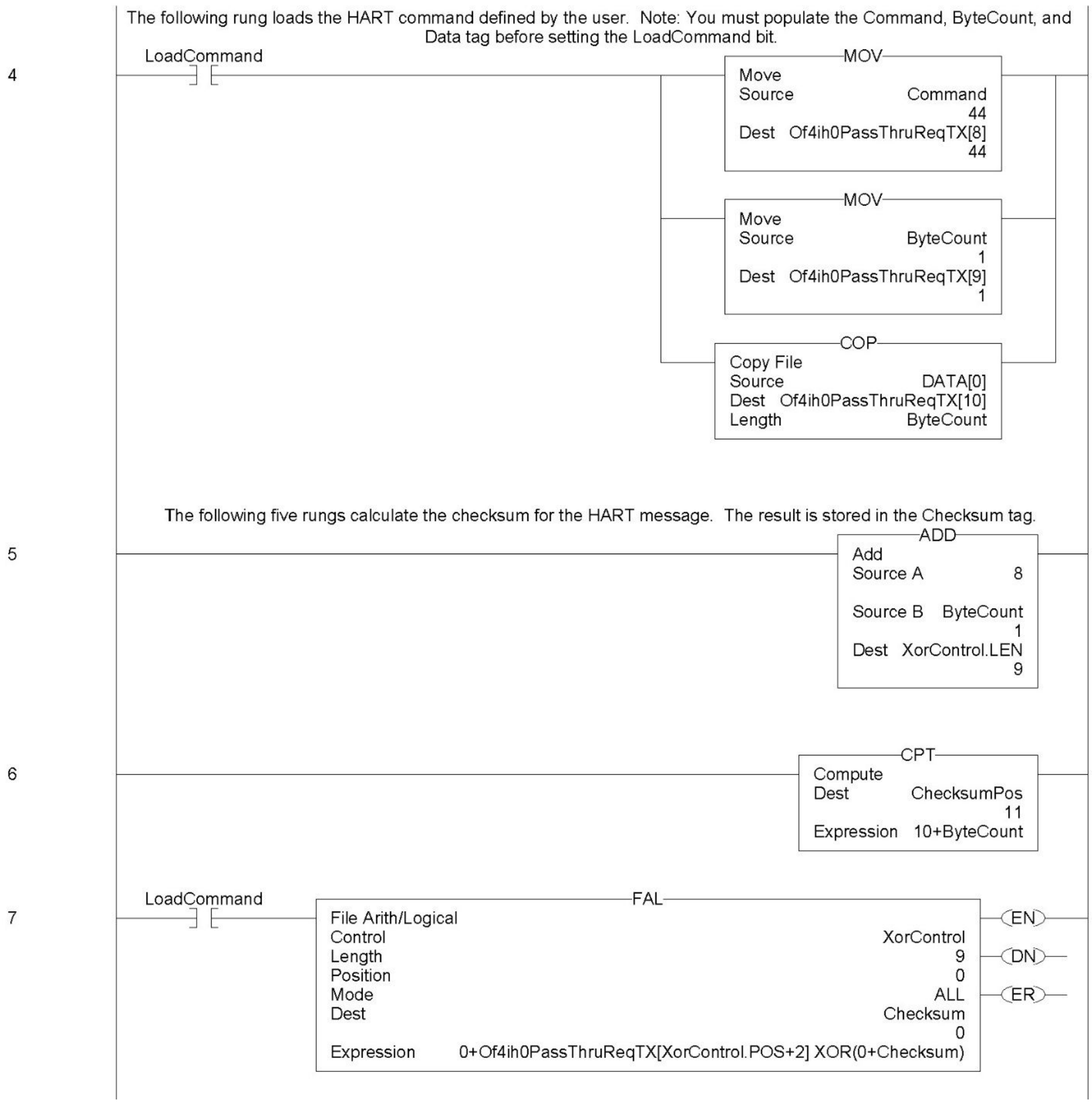

The following rung loads the HART command defined by the user. Note: You must populate the Command, ByteCount, and Data tag before setting the LoadCommand bit.
4
LoadCommand
MOV
Move
Source Command
44
Dest Of4ih0PassThruReqTX[8]
44
MOV
Move
Source ByteCount
1
Dest Of4ih0PassThruReqTX[9]
1
COP
Copy File
Source DATA[0]
Dest Of4ih0PassThruReqTX[10]
Length ByteCount
The following five rungs calculate the checksum for the HART message. The result is stored in the Checksum tag.
5
ADD
Add
Source A 8
Source B ByteCount
1
Dest XorControl.LEN
9
6
CPT
Compute
Dest ChecksumPos
11
Expression 10+ByteCount
7
LoadCommand
FAL
File Arith/Logical
Control XorControl
Length 9
Position 0
Mode ALL
Dest Checksum
0
Expression 0+Of4ih0PassThruReqTX[XorControl.POS+2] XOR(0+Checksum)
EN
DN
ER

Figure 6-10d.

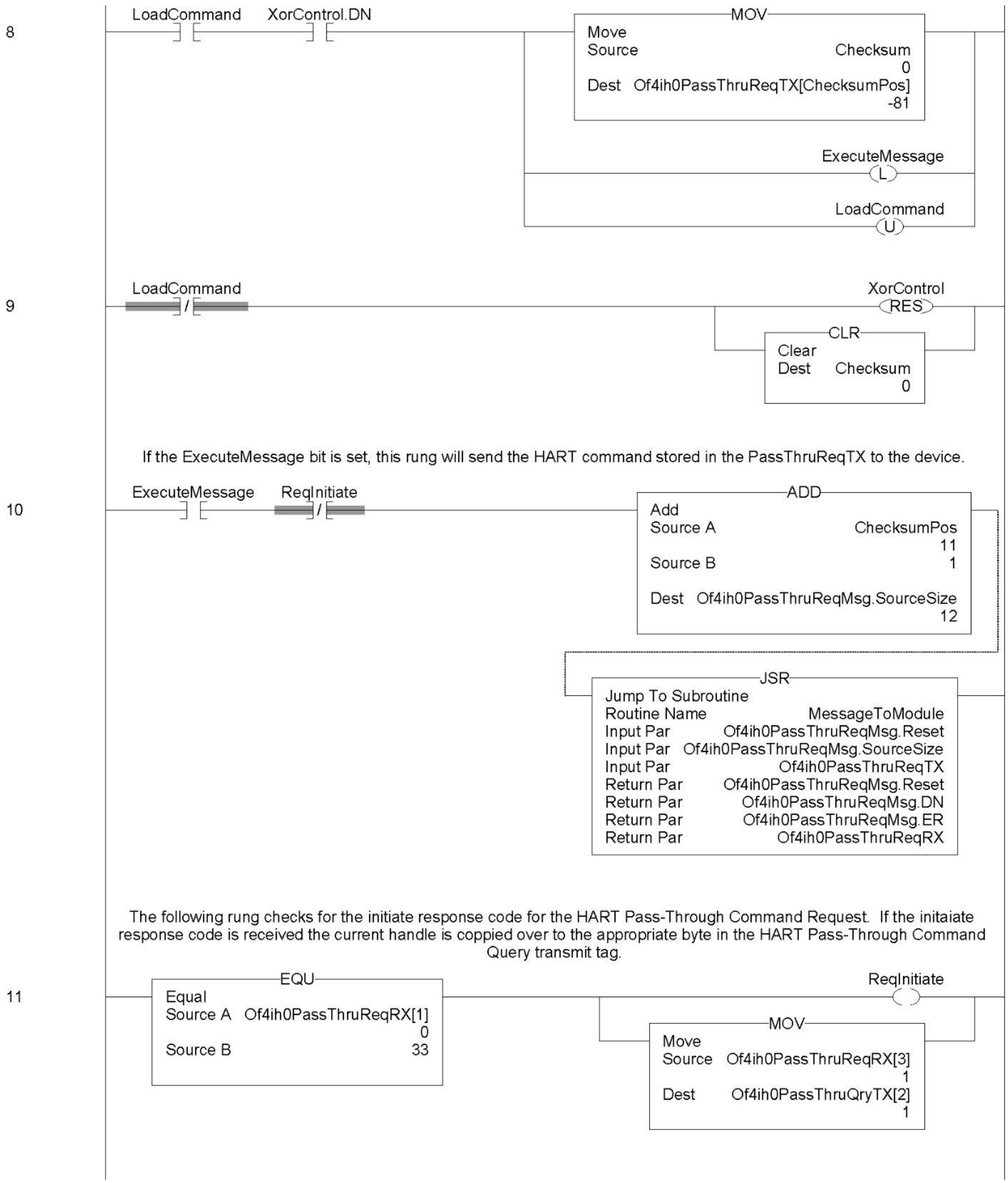

8
LoadCommand
XorControl.DN
MOV
Move
Source Checksum
0
Dest Of4ih0PassThruReqTX[ChecksumPos]
-81
ExecuteMessage
L
LoadCommand
U
9
LoadCommand
XorControl
RES
CLR
Clear
Dest Checksum
0
If the ExecuteMessage bit is set, this rung will send the HART command stored in the PassThruReqTX to the device.
10
ExecuteMessage
ReqInitiate
ADD
Add
Source A ChecksumPos
11
Source B 1
Dest Of4ih0PassThruReqMsg.SourceSize
12
JSR
Jump To Subroutine
Routine Name MessageToModule
Input Par Of4ih0PassThruReqMsg.Reset
Input Par Of4ih0PassThruReqMsg.SourceSize
Input Par Of4ih0PassThruReqTX
Return Par Of4ih0PassThruReqMsg.Reset
Return Par Of4ih0PassThruReqMsg.DN
Return Par Of4ih0PassThruReqMsg.ER
Return Par Of4ih0PassThruReqRX
The following rung checks for the initiate response code for the HART Pass-Through Command Request. If the initaiate response code is received the current handle is coppied over to the appropriate byte in the HART Pass-Through Command Query transmit tag.
11
EQU
Equal
Source A Of4ih0PassThruReqRX[1]
0
Source B 33
ReqInitiate
MOV
Move
Source Of4ih0PassThruReqRX[3]
1
Dest Of4ih0PassThruQryTX[2]
1

Figure 6-10e.

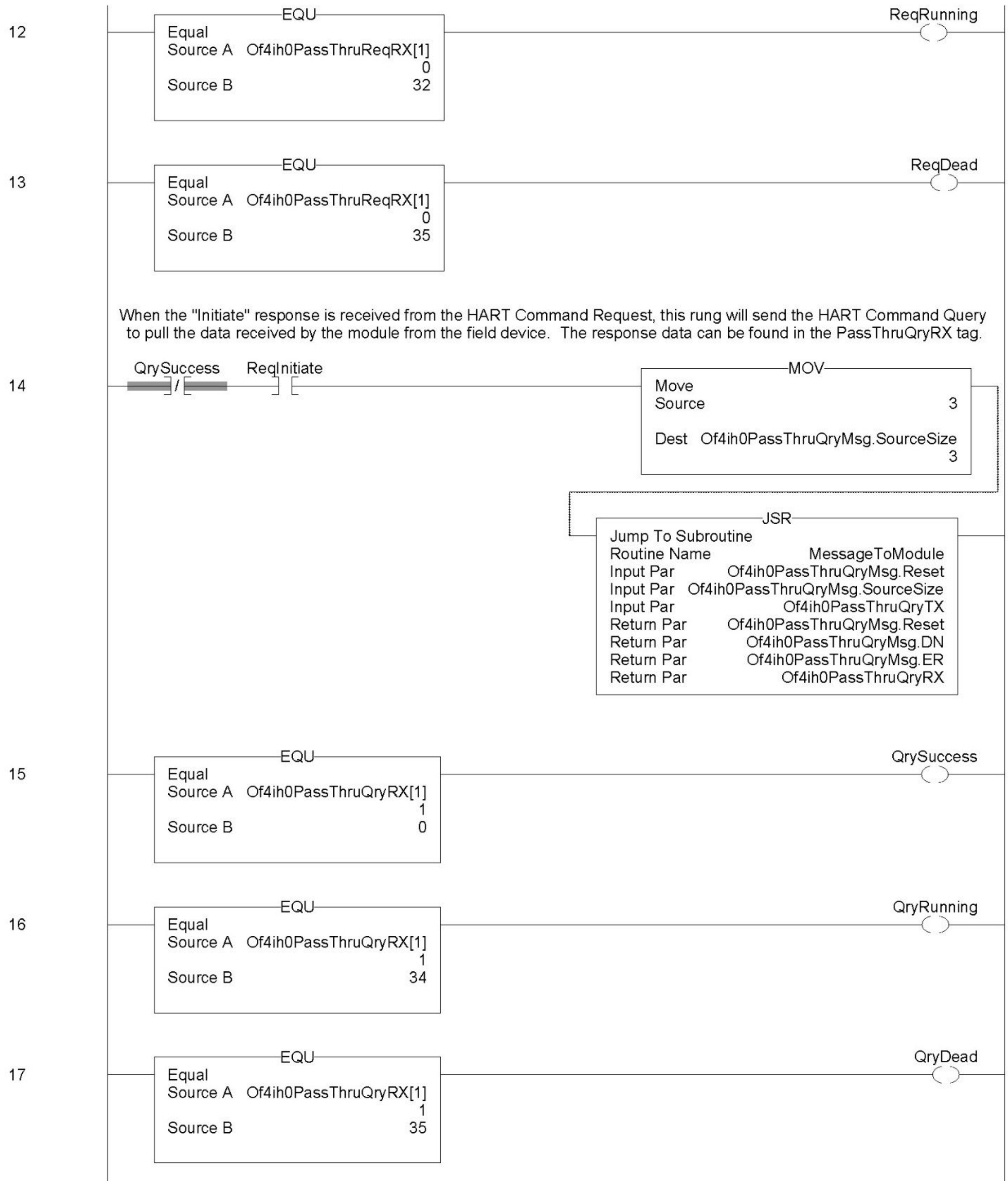
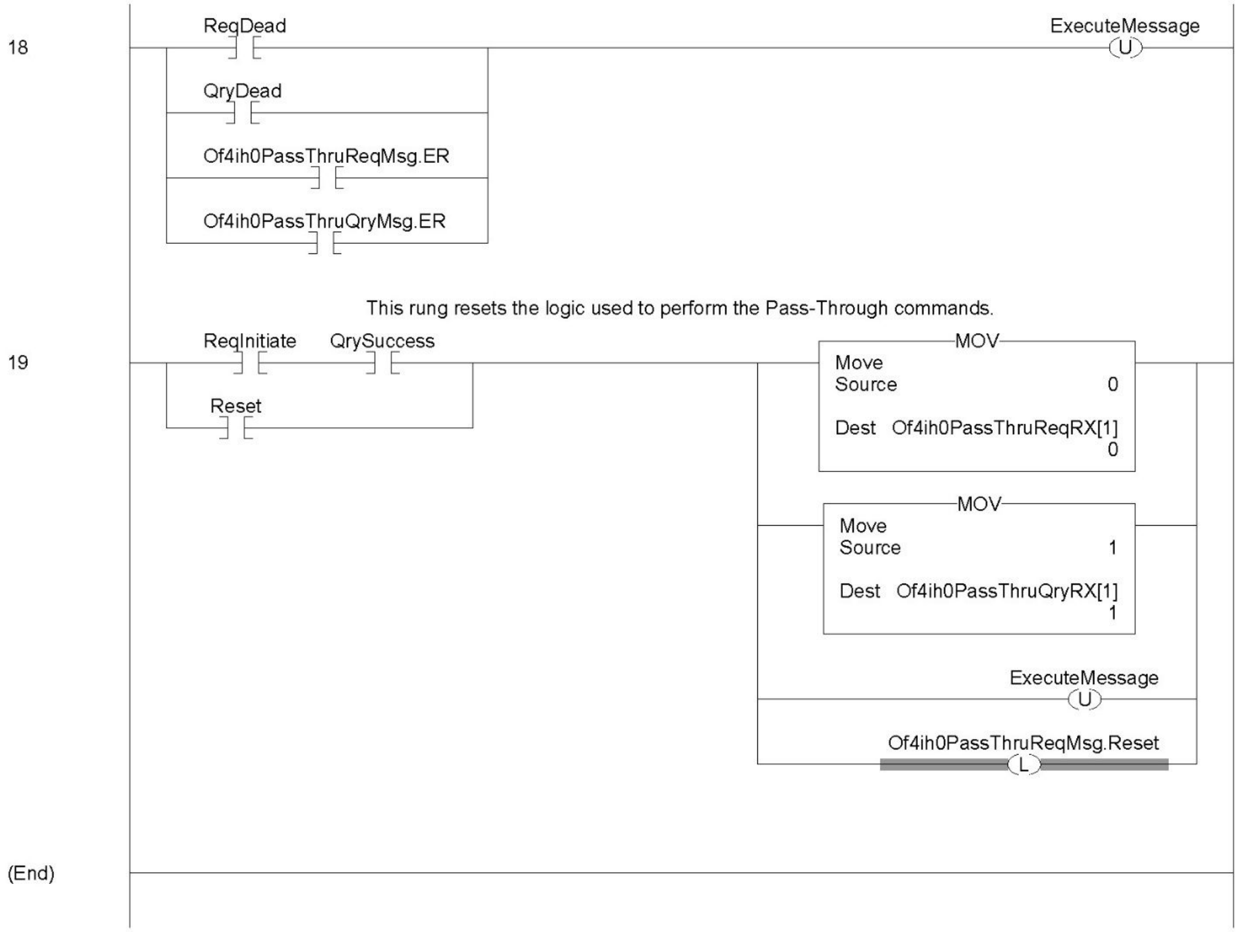

12
EQU
Equal
Source A Of4ih0PassThruReqRX[1]
0
Source B 32
ReqRunning
13
EQU
Equal
Source A Of4ih0PassThruReqRX[1]
0
Source B 35
ReqDead
When the "Initiate" response is received from the HART Command Request, this rung will send the HART Command Query to pull the data received by the module from the field device. The response data can be found in the PassThruQryRX tag.
14
QrySuccess
ReqInitiate
MOV
Move
Source 3
Dest Of4ih0PassThruQryMsg.SourceSize
3
JSR
Jump To Subroutine
Routine Name MessageToModule
Input Par Of4ih0PassThruQryMsg.Reset
Input Par Of4ih0PassThruQryMsg.SourceSize
Input Par Of4ih0PassThruQryTX
Return Par Of4ih0PassThruQryMsg.Reset
Return Par Of4ih0PassThruQryMsg.DN
Return Par Of4ih0PassThruQryMsg.ER
Return Par Of4ih0PassThruQryRX
15
EQU
Equal
Source A Of4ih0PassThruQryRX[1]
1
Source B 0
QrySuccess
16
EQU
Equal
Source A Of4ih0PassThruQryRX[1]
1
Source B 34
QryRunning
17
EQU
Equal
Source A Of4ih0PassThruQryRX[1]
1
Source B 35
QryDead

Figure 6-10f.



NOTE	The ladder in Figure 6-10 can be found in the project sample file located on our website at (www.spectrumcontrols.com)

Section 6.5 HART Protocol Overview

To read and write HART commands to, and from, the field device reliably using the OF4IH, you must have a basic knowledge of the HART protocol. This section explains the various segments that make up the HART message and how to formulate the message and send it to the field device using the module-specific Pass-Through command, which was described earlier in this chapter.

6.5.1 Message Format

HART protocol specifies a message structure as follows:

Preamble	Start Character	Address	Command	Byte Count	Status	Data	Checksum

NOTE	The HART protocol supports two different formats, long and short frame. Older HART instruments (up to HART revision 4) used a short frame format. In this format, the address of the slave device is either 0, for non-multidrop devices using the 4-20 mA current signal, or 1-15 for multidrop devices.

HART revision 5 introduced the long frame format. In this format, the address of a slave device is a worldwide, unique 38-bit number derived from the manufacturer code, the device type code, and the device identification number. The long frame format provides extra security against acceptance of commands meant for other devices, due to external interference or excessive crosstalk. The OF4IH supports only the long frame format.

Each item of the message structure shown above is explained as follows.

Preamble

The preamble consists of three or more hexadecimal FF characters (all 1s) allowing the receiving modem to get its frequency-detection circuits synchronized to the signal after any pause in transmission.

NOTE	The preamble does not need to be included in the HART message when using the module specific Pass-through command. The Pass-through command already includes the preamble.

Start Character

The start character in a HART message has various values, indicating which frame format is being used, the source of the message, and whether a field device is in burst mode. The possible definitions are shown in the table below.

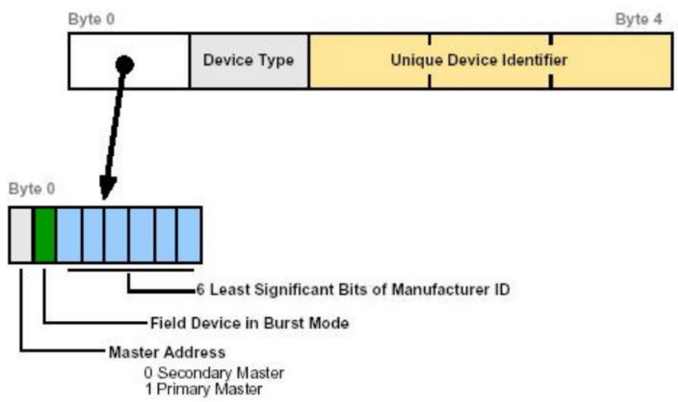
Table 6-14. Start Character Definition

	Short Frame	Long Frame
Master to slave	02 (Hex)	82 (Hex)
Slave to master	06 (Hex)	86 (Hex)
Burst mode from slave	01 (Hex)	81 (Hex)

Address

The address field contains both the host and field device addresses for the message. These may be contained in a single byte (short frame format) or in five bytes (long frame format). Since the module presently only supports the long frame form, we will omit the discussion of the short frame form. In either format, the single-bit address of the master is the most significant. Only two masters are allowed for example, a control system and a hand-held communicator. The most significant bit of the address field differentiates these two hosts. Primary masters such as the OF4IH use address 1, and secondary masters such as handhelds use address 0. Please see figure below.

Figure 6-11. Long Frame Address



NOTE	The OF4IH does not support burst mode.

The 1-byte Device Type code is allocated and controlled by the manufacturer. The 3-byte Device Identifier is similar to a serial number in that each device manufactured with the same Device Type Code must have a different Device Identifier. The OF4IH automatically pulls for the device-specific codes using the Auto-acquisition process. The device-specific codes that are acquired using this process can be seen in Table 6-2.

Command

The command byte contains an integer (0 to hex FF or decimal 257) that represents one of the HART commands. Code 254 is defined as an expansion code and is followed by another byte allowing more than 256 different commands to be defined if necessary. The received command code is echoed back by the slave device in its reply.

There are three categories of commands: universal commands, which all HART devices must implement; common practice commands, which should be used if the associated function is provided; and device-specific commands, which are for functions unique to a specific slave device.

Byte Count

The byte count portion of the message contains an integer value representing the number of bytes that form the remainder of this message excluding the checksum. In other words, the byte count determines the length of the data and status.

Status

Status is included only in reply messages from a slave. It consists of two bytes of bit-coded information. The first byte indicates communication errors, if any. Otherwise, if communication was good, this byte may indicate the status of the received command such as a busy device, or a command not recognized. The second status byte indicates the operational state of the slave device. A properly operating slave device will have both status bytes set to logical zero. The meaning of the individual status bits can be found in Appendix B.

Data

This portion of the HART message contains the data, if any, for the command. Not all commands or responses contain data. For those that do, up to 25 bytes can be included. Data may be in the form of unsigned integers, floating point numbers, or ASCII character strings. The number of bytes of data and the data format used for each item are specified for each HART command.

Checksum

The checksum byte contains the exclusive-or (longitudinal parity) of all the bytes that precede it in the message starting with the Start Character. This provides a further check on transmission integrity, beyond the parity check on the 8 bits of each individual byte.

6.5.2 Sending a HART Command to a Field Device via Pass-through

The next step is to formulate a message and successfully send the message to the field device using the pass-through command. The first step is to formulate the message and populate the source tag Of4ih0PassThruReqTX. This tag is used in the ladder sample provided.

Table 6-15. Source Tag Information

Tag Name	Value in Hex	Description
HART_PASS_THRU_QRY_RX[0]	00	HART channel
HART_PASS_THRU_QRY_RX[1]	00	Command Status
HART_PASS_THRU_QRY_RX[2]	15	Length of handle + HART response data (Byte 1)
HART_PASS_THRU_QRY_RX[3]	0	Length of handle + HART response data (Byte 2)
HART_PASS_THRU_QRY_RX[4]	02	Message handle
HART_PASS_THRU_QRY_RX[5]	86	Start character
HART_PASS_THRU_QRY_RX[6]	BE	Long address byte 0
HART_PASS_THRU_QRY_RX[7]	02	Long address byte 1
HART_PASS_THRU_QRY_RX[8]	0C	Long address byte 2
HART_PASS_THRU_QRY_RX[9]	77	Long address byte 3
HART_PASS_THRU_QRY_RX[10]	37	Long address byte 4
HART_PASS_THRU_QRY_RX[11]	23	HART command = 35 decimal
HART_PASS_THRU_QRY_RX[12]	0B	Byte count = 11 decimal
HART_PASS_THRU_QRY_RX[13]	00	Status Byte 0
HART_PASS_THRU_QRY_RX[14]	00	Status Byte 1
HART_PASS_THRU_QRY_RX[15]	20	Range units code = 32 decimal
HART_PASS_THRU_QRY_RX[16]	44	Upper Range value (This is a floating-point value = 600) Note: The bytes are in reverse order.
HART_PASS_THRU_QRY_RX[17]	16	
HART_PASS_THRU_QRY_RX[18]	00	
HART_PASS_THRU_QRY_RX[19]	00	
HART_PASS_THRU_QRY_RX[20]	C3	Lower Range value (This is a floating-point value = -150) Note: The bytes are in reverse order.
HART_PASS_THRU_QRY_RX[21]	16	
HART_PASS_THRU_QRY_RX[22]	00	
HART_PASS_THRU_QRY_RX[23]	00	
HART_PASS_THRU_QRY_RX[24]	F9	Checksum

The HART message string, shown above, performs HART command 35 (write range values). Once the tags are populated with the HART message, the message can be sent using the ladder provided. The reply for the HART command will be found in the Of4ih0PassThruQryRX tag. The response message should look like the table shown below.

Table 6-16. Response Tag Information

Tag Name	Value in Hex	Description
HART_PASS_THRU_REQ_TX[0]	00	HART channel
HART_PASS_THRU_REQ_TX[1]	01	Pass-through command designator
HART_PASS_THRU_REQ_TX[2]	82	Start character
HART_PASS_THRU_REQ_TX[3]	BE	Long address byte 0
HART_PASS_THRU_REQ_TX[4]	02	Long address byte 1
HART_PASS_THRU_REQ_TX[5]	0C	Long address byte 2
HART_PASS_THRU_REQ_TX[6]	77	Long address byte 3
HART_PASS_THRU_REQ_TX[7]	37	Long address byte 4
HART_PASS_THRU_REQ_TX[8]	23	HART command = 35 decimal
HART_PASS_THRU_REQ_TX[9]	09	Byte count
HART_PASS_THRU_REQ_TX[10]	20	Range units code = 32 decimal
HART_PASS_THRU_REQ_TX[11]	44	Upper Range value (This is a floating-point value = 600.0) Note: The bytes are in reverse order.
HART_PASS_THRU_REQ_TX[12]	16	
HART_PASS_THRU_REQ_TX[13]	00	
HART_PASS_THRU_REQ_TX[14]	00	
HART_PASS_THRU_REQ_TX[15]	C3	Lower Range value (This is a floating-point value = -150.0) Note: The bytes are in reverse order.
HART_PASS_THRU_REQ_TX[16]	16	
HART_PASS_THRU_REQ_TX[17]	00	
HART_PASS_THRU_REQ_TX[18]	00	
HART_PASS_THRU_REQ_TX[19]	FF	Checksum

Chapter 7
Programming Examples

This chapter provides ladder samples for general and advanced applications using the 1756sc-OF4IH module. Ladder samples for both the CompactLogix and MicroLogix 1500 PLC are discussed in this chapter.

Section 7.1 CompactLogix

7.1.1 Reset/Reconfig

The following rungs of ladder demonstrate how to either reset the module (Backplane connection will be broken) or reconfigure the module without breaking the connection.

Figure 7-1. Reset

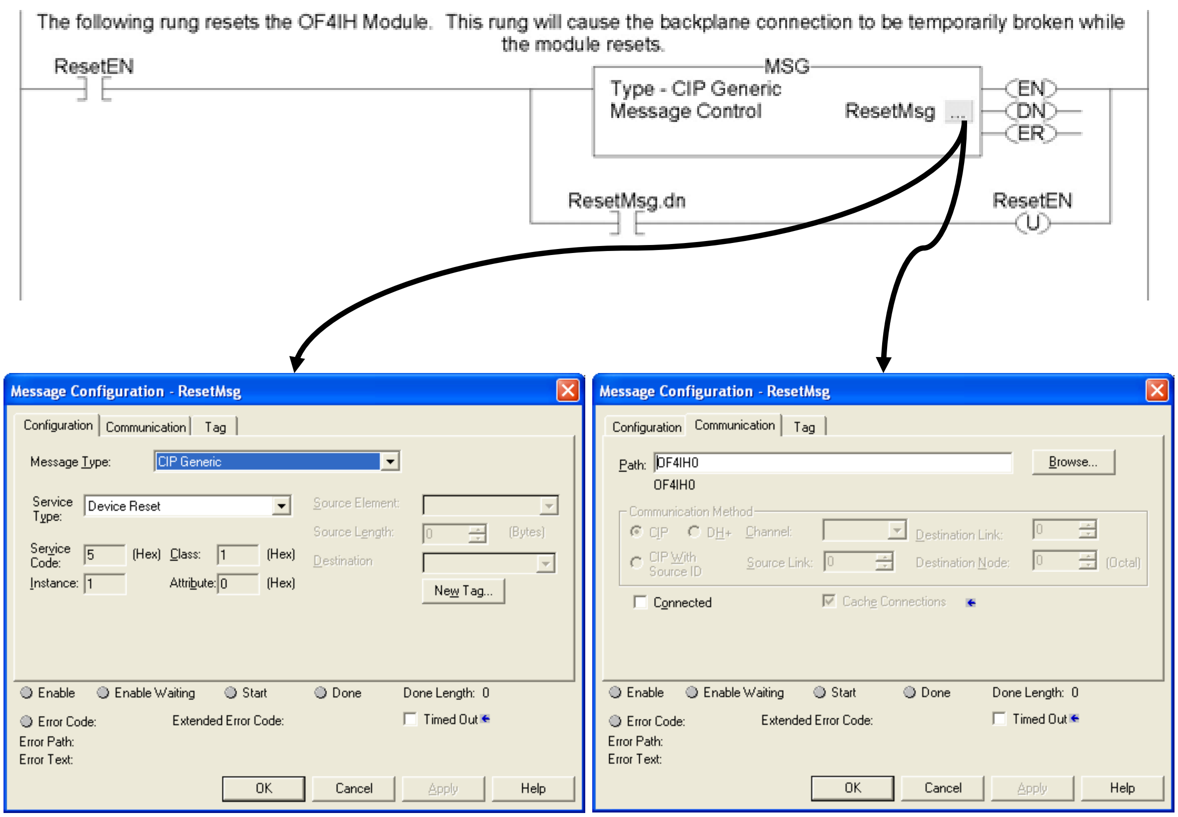
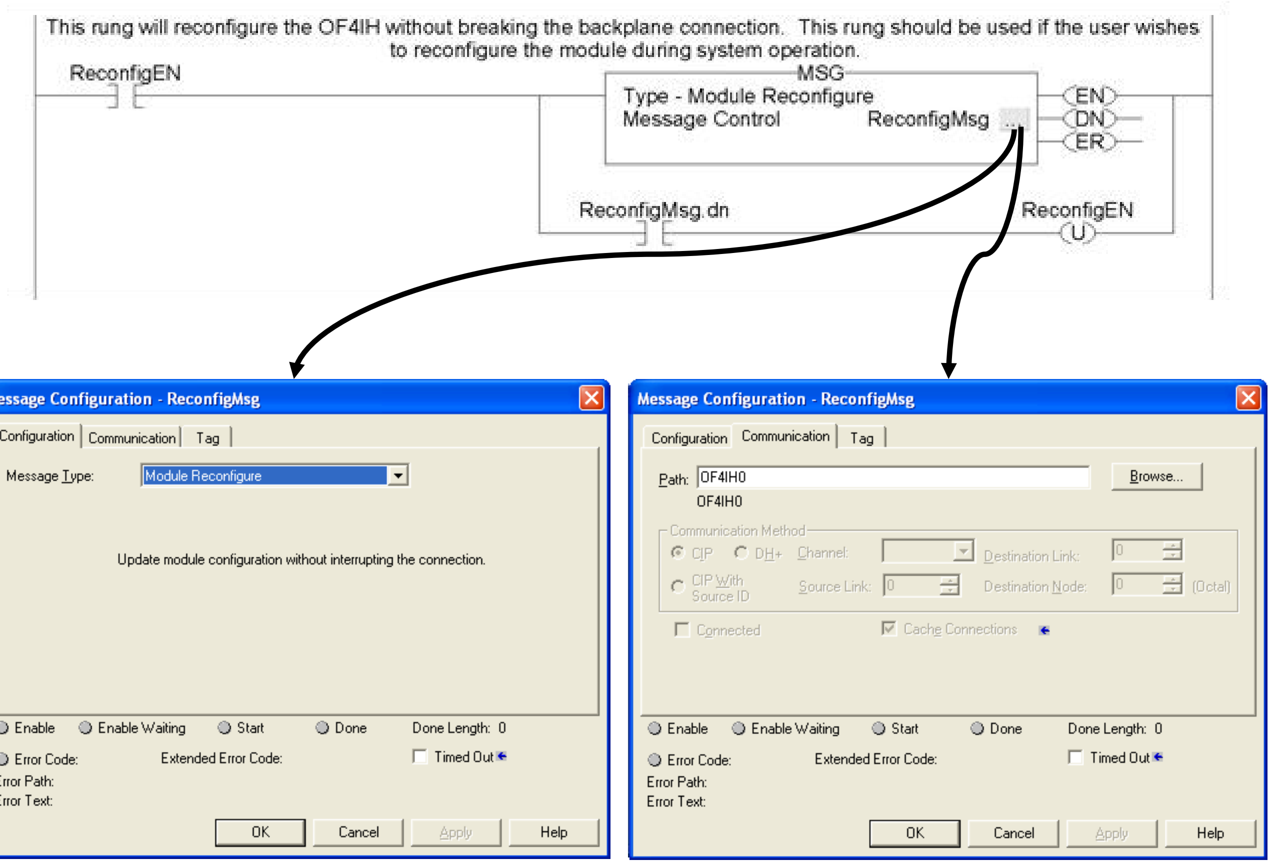


Figure 7-2. Reconfig

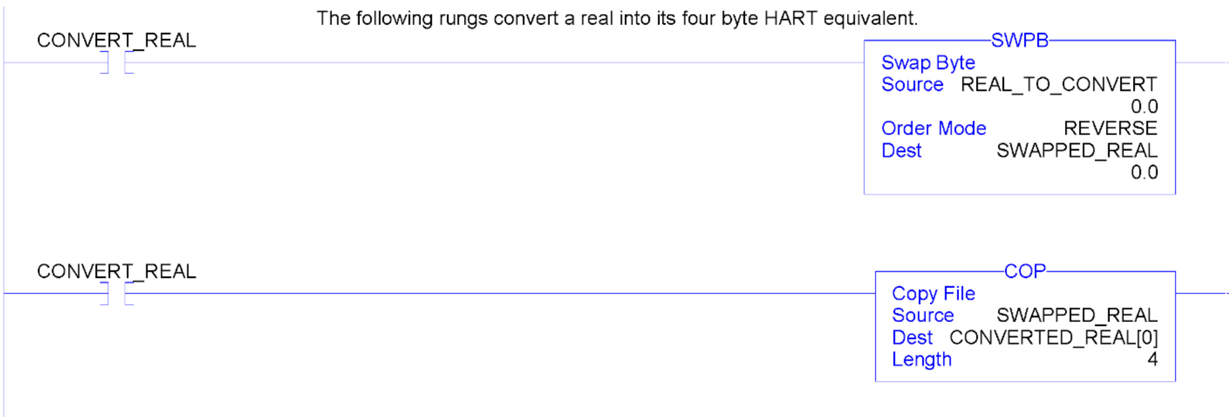


7.1.2 Swap Byte Order

This ladder sample demonstrates how to reverse the order of the bytes for a floating-point tag and then convert it to 4 consecutive SINT tags, so that it can be used in a HART message.

NOTE	If the HART message being sent or received using the pass-through command contains floating point values, the order of the bytes must be reversed.

Figure 7-3. Change Byte Order



7.1.3 Converting Unpacked ASCII to Packed ASCII

Packed ASCII is a HART-specific, 6-bit character code representing a subset of the ASCII character code set (see table below). Produced by compressing four packed ASCII characters into three 8-bit bytes, packed ASCII strings must be a multiple of 4 characters (3 bytes) and must be padded out to the end of the data item with space characters. For example, 4 space characters at the end of a string would appear as the 3 bytes: 0×82, 0×08 and 0×20.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?

Note: Most significant hexadecimal digit top to bottom; least significant left to right.

Construction of Packed ASCII characters:

Constructing a packed ASCII string is a simple matter of discarding the most significant two bits from each character and compressing the result:

1. Truncate Bits 6 and 7 of each ASCII character.
2. Pack four, 6 bit-ASCII characters into three bytes.
3. Repeat until the entire string is processed.

The algorithm can be implemented in ladder by masking and shifting four 6-bit characters into a double word register then moving the three bytes into the packed ASCII string.

Reconstruction of ASCII characters:

Unpacking packed ASCII strings requires flipping some bits in addition to uncompressing the string itself. To unpack a packed ASCII string:

1. Unpack the four, 6-bit ASCII characters.
2. For each character, place the complement of bit 5 into bit 6.
3. For each character, reset bit 7 to zero.
4. Repeat until the entire string is processed.

This algorithm can be implemented by loading three bytes into a 24-bit register and shifting the four 6-bit characters into the string. Parse the resulting character to flip bit 6 as needed.

The ladder sample starting on the next page demonstrates how to pack 4 unpacked ASCII characters into 3 bytes.

Figure 7-4a. Packed ASCII

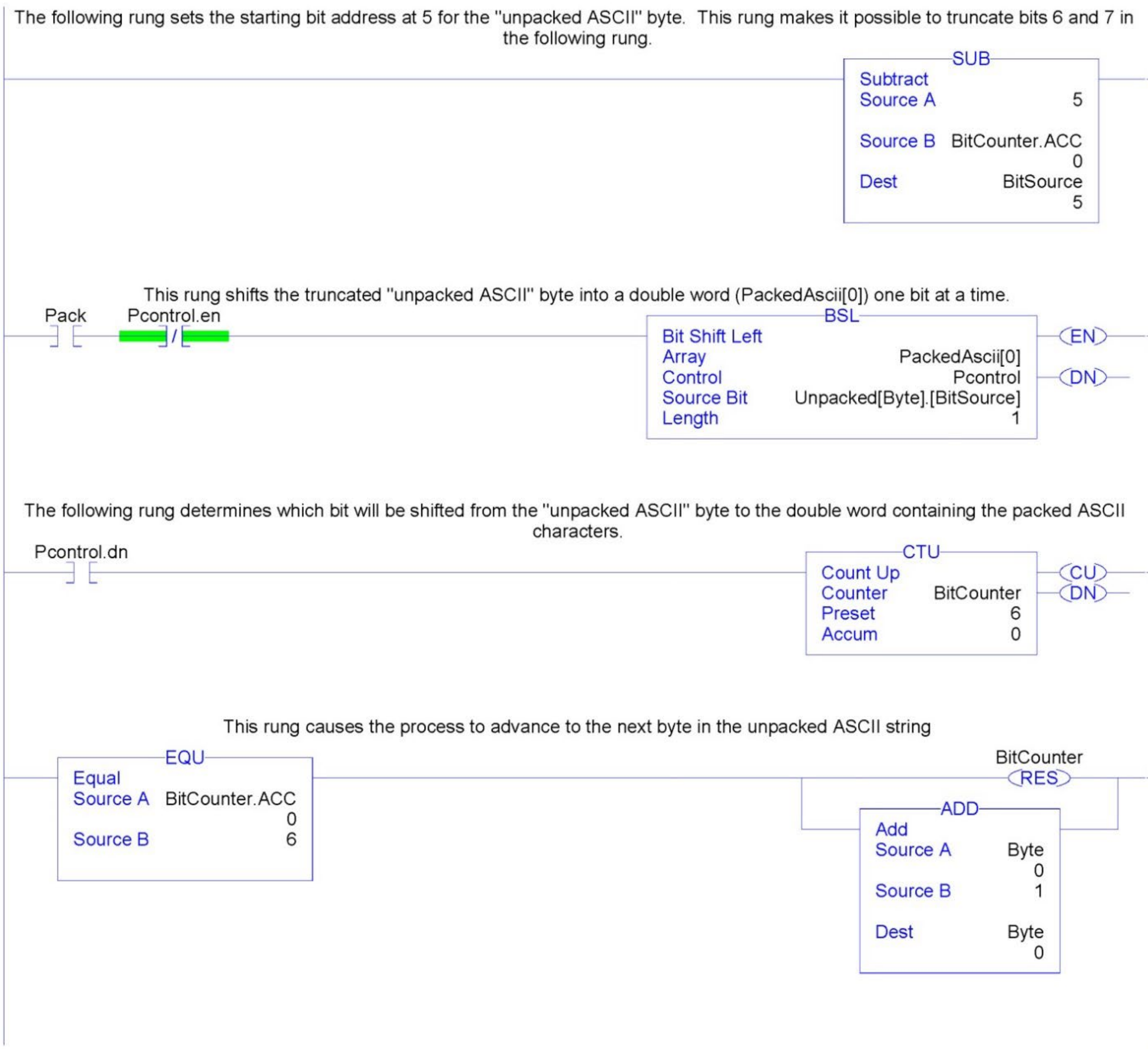
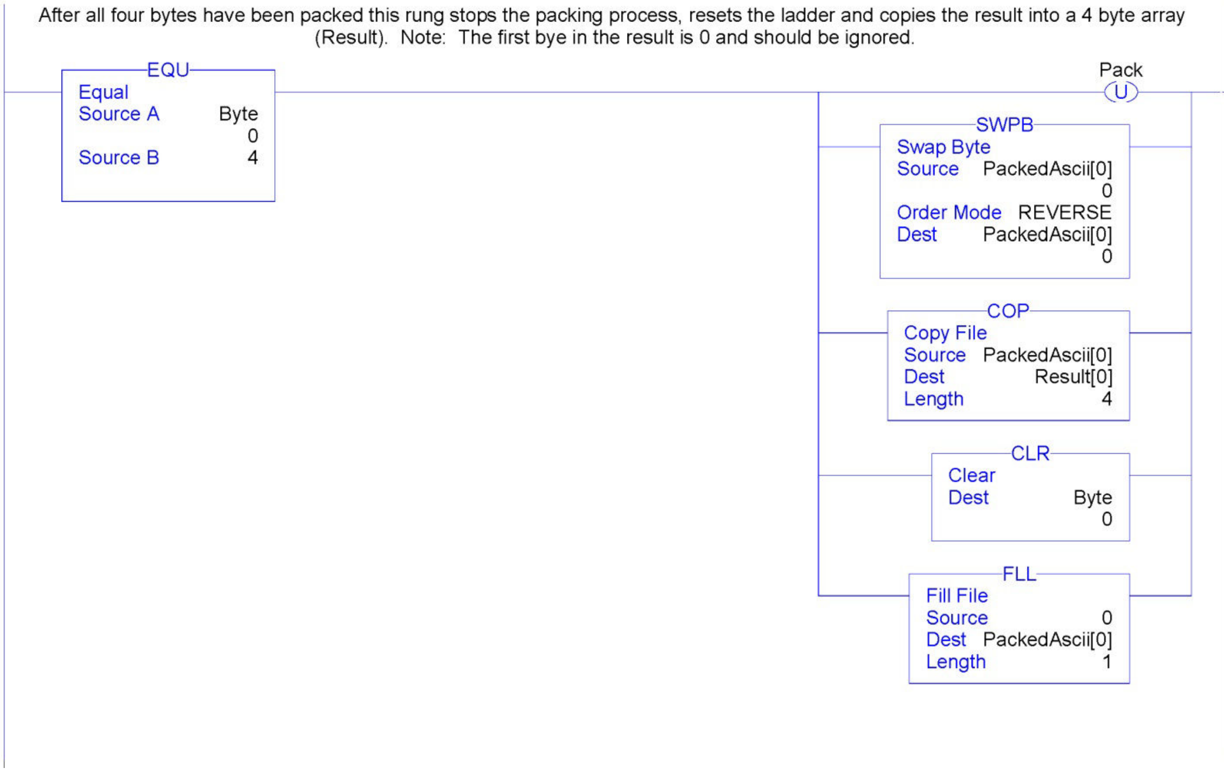


Figure 7-4b.


After all four bytes have been packed this rung stops the packing process, resets the ladder and copies the result into a 4 byte array (Result). Note: The first bye in the result is 0 and should be ignored.
EQU
Equal
Source A Byte
0
Source B 4
Pack
U
SWPB
Swap Byte
Source PackedAscii[0]
0
Order Mode REVERSE
Dest PackedAscii[0]
0
COP
Copy File
Source PackedAscii[0]
Dest Result[0]
Length 4
CLR
Clear
Dest Byte
0
FLL
Fill File
Source 0
Dest PackedAscii[0]
Length 1

Section 7.2 MicroLogix 1500

The following ladder samples provide a working HART solution for the MicroLogix 1500 when used with the OF4IH module. The following table briefly describes each routine in the project file.

NOTE	It is recommended that a 1764-LRP series C processor with firmware version 5 or higher be used. The LRP processor supports floating point files, which is required to read floating point data from the OF4IH.

Table 7-1. Routine Description

Routine	Description
MAIN	The main routine is the starting point for the ladder program.
PACKETS	The "packets" routine is used to demultiplex the HART data from the input file to individual integer files, so that the data can be viewed or used within the ladder program. This routine is called from the MAIN routine.
MSG_TO_MOD	This routine is used to send and receive messages to, and from, the module. Refer to Chapter 6 for more details regarding sending and receiving messages. This routine is called from the HART_MSG routine.
SRC_CHECK	Calculates the checksum for a message sent to the module one page at a time. This routine is called from the MSG_TO_MOD routine.
DEST_CHECKSUM	This routine calculates the checksum for a message received from the module one page at a time. This routine is called from the MSG_TO_MOD routine.
HART_MSG	This routine composes HART messages that will be sent to the module/field transmitter. This routine is called from the MAIN routine.
WORD_BYTE	Converts word data to its byte equivalent. This routine is called from the HART_MSG routine.
HART_CHECK	Calculates the checksum for the HART message being sent to the module/field device. This routine is called from the HART_MSG routine.
BYTE_WORD	Converts byte data to its word equivalent. This routine is called by the HART_MSG routine.

7.2.1 MAIN Routine

The main routine is the starting point for the ladder program.

Figure 7-5. Main Routine

7.2.2 PACKETS Routine

The “packets” routine is used to demultiplex the HART data from the input file to individual integer files, so that the data can be viewed or used within the ladder program.   This routine is called from the MAIN routine.

Figure 7-6a. Packets Routine

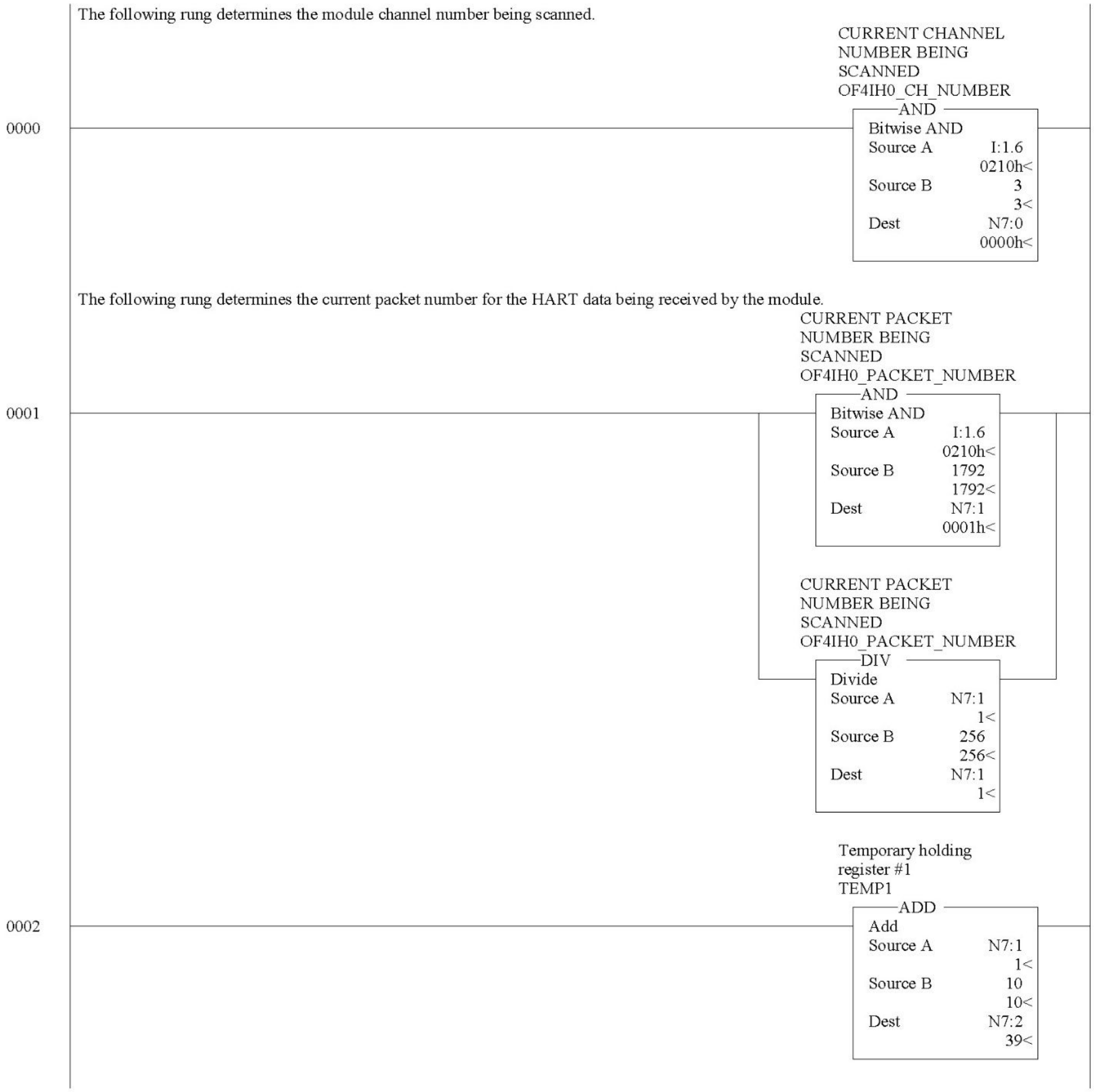


Figure 7-6b.

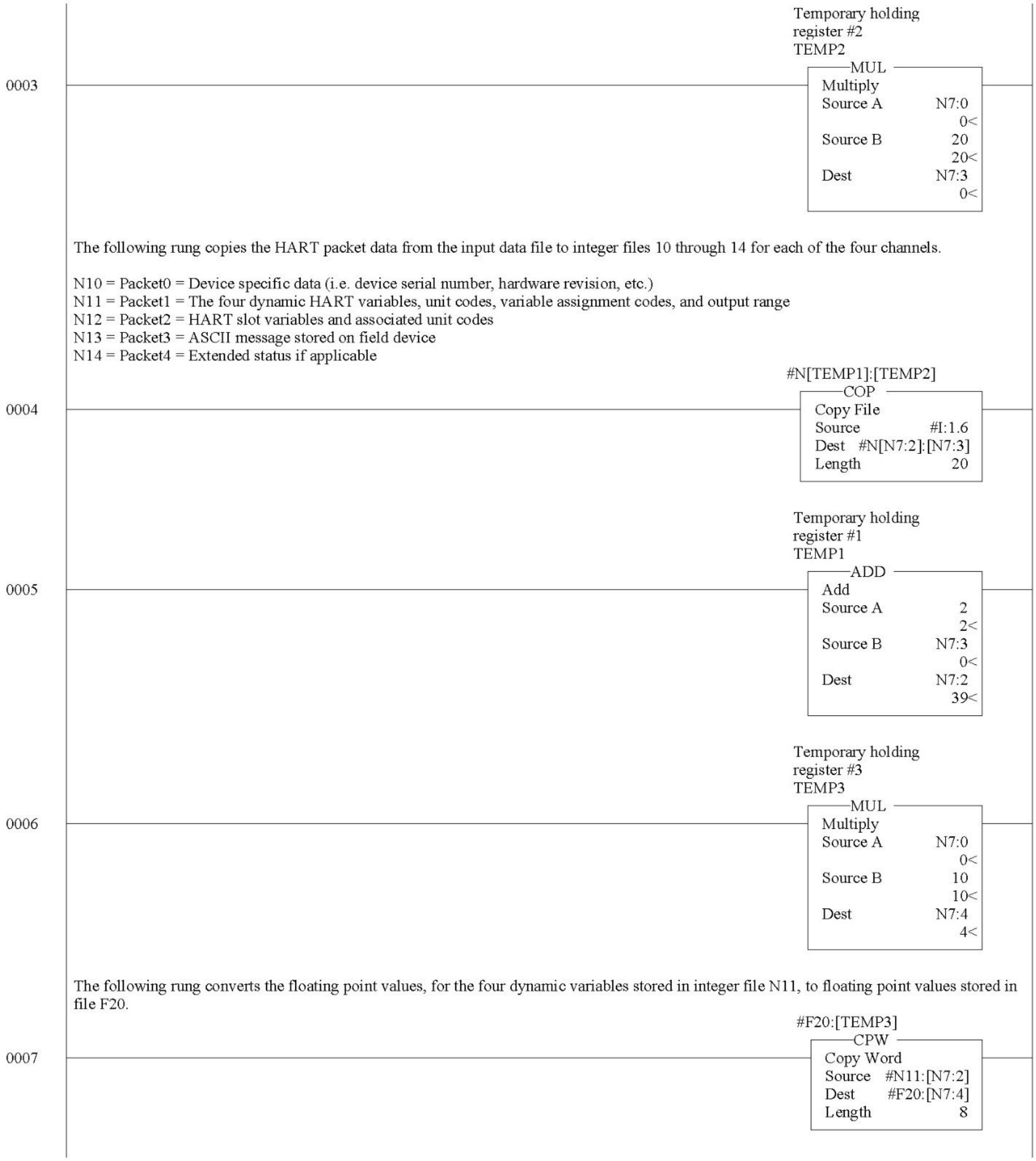
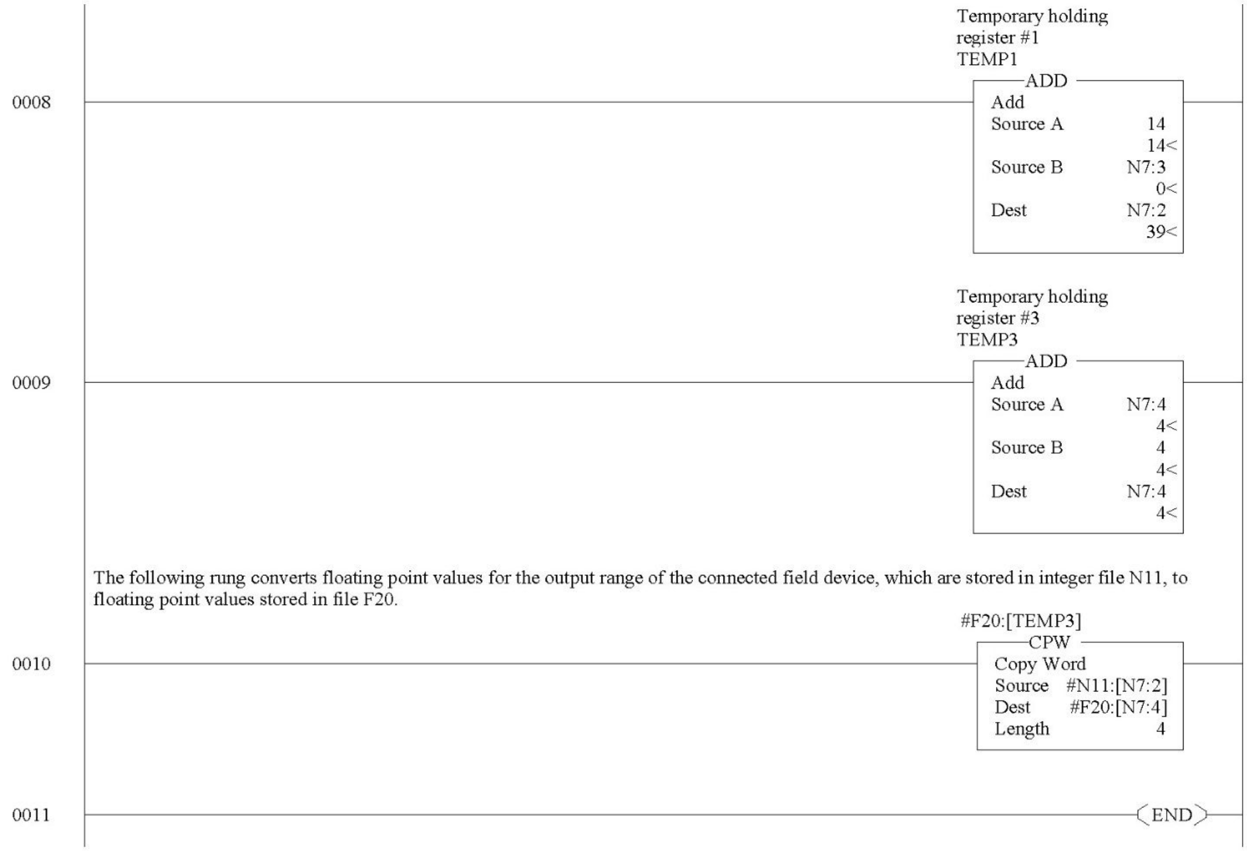

Temporary holding
register #2
TEMP2
0003
MUL
Multiply
Source A N7:0
0<
Source B 20
20<
Dest N7:3
0<
The following rung copies the HART packet data from the input data file to integer files 10 through 14 for each of the four channels.
N10 = Packet0 = Device specific data (i.e. device serial number, hardware revision, etc.)
N11 = Packet1 = The four dynamic HART variables, unit codes, variable assignment codes, and output range
N12 = Packet2 = HART slot variables and associated unit codes
N13 = Packet3 = ASCII message stored on field device
N14 = Packet4 = Extended status if applicable
#N[TEMP1]:[TEMP2]
0004
COP
Copy File
Source #I:1.6
Dest #N[N7:2]:[N7:3]
Length 20
Temporary holding
register #1
TEMP1
0005
ADD
Add
Source A 2
2<
Source B N7:3
0<
Dest N7:2
39<
Temporary holding
register #3
TEMP3
0006
MUL
Multiply
Source A N7:0
0<
Source B 10
10<
Dest N7:4
4<
The following rung converts the floating point values, for the four dynamic variables stored in integer file N11, to floating point values stored in file F20.
#F20:[TEMP3]
0007
CPW
Copy Word
Source #N11:[N7:2]
Dest #F20:[N7:4]
Length 8

Figure 7-6c.


Temporary holding
register #1
TEMP1
ADD
Add
Source A 14
14<
Source B N7:3
0<
Dest N7:2
39<
0008
Temporary holding
register #3
TEMP3
ADD
Add
Source A N7:4
4<
Source B 4
4<
Dest N7:4
4<
0009
The following rung converts floating point values for the output range of the connected field device, which are stored in integer file N11, to floating point values stored in file F20.
#F20:[TEMP3]
CPW
Copy Word
Source #N11:[N7:2]
Dest #F20:[N7:4]
Length 4
0010
0011
END

7.2.3 MSG_TO_MOD Routine

This routine is used to send and receive messages to and from the module. Refer to Chapter 7 for more details regarding sending and receiving messages. This routine is called from the HART_MSG routine.

Figure 7-7a. Message to Module

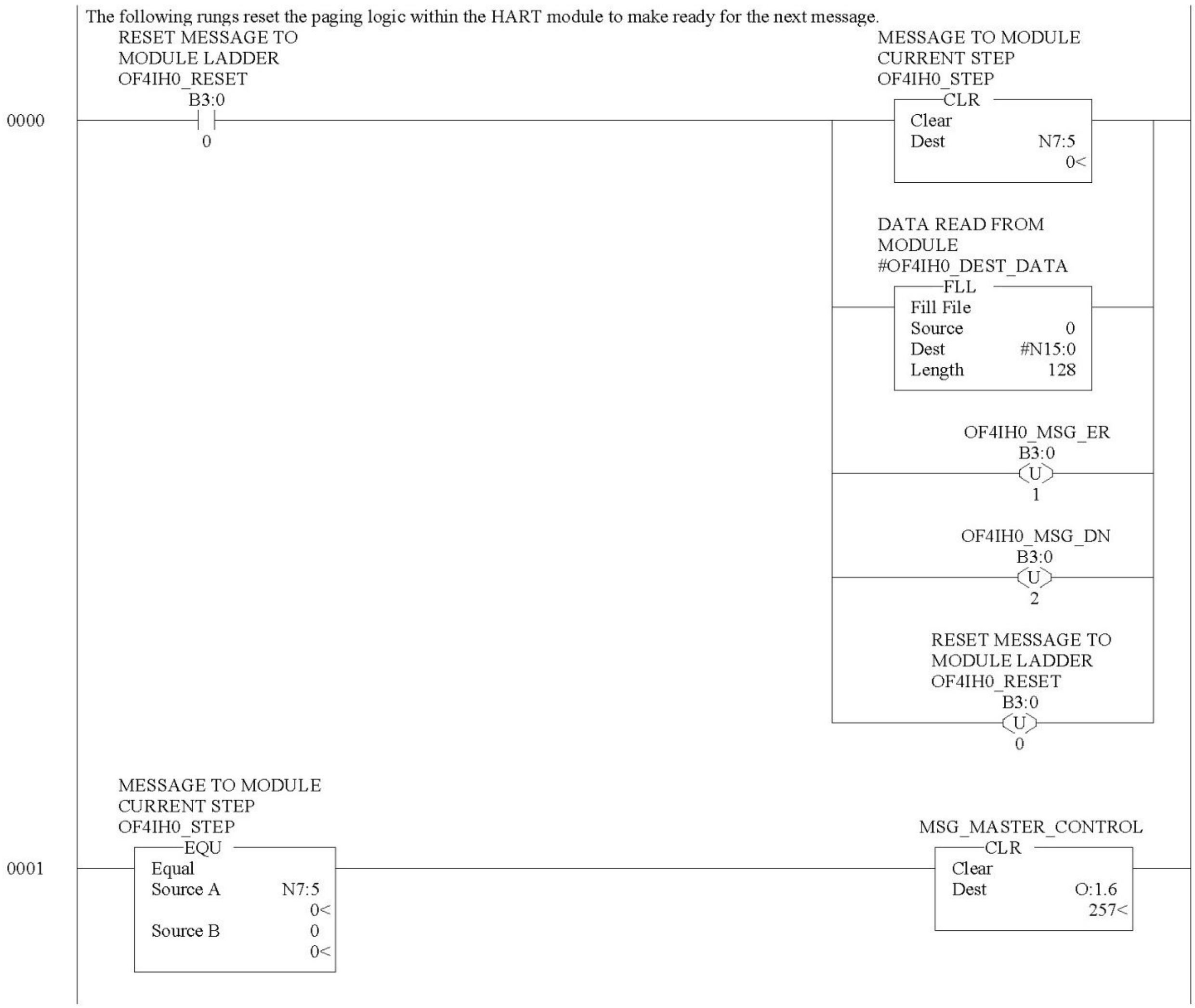


Figure 7-7b.

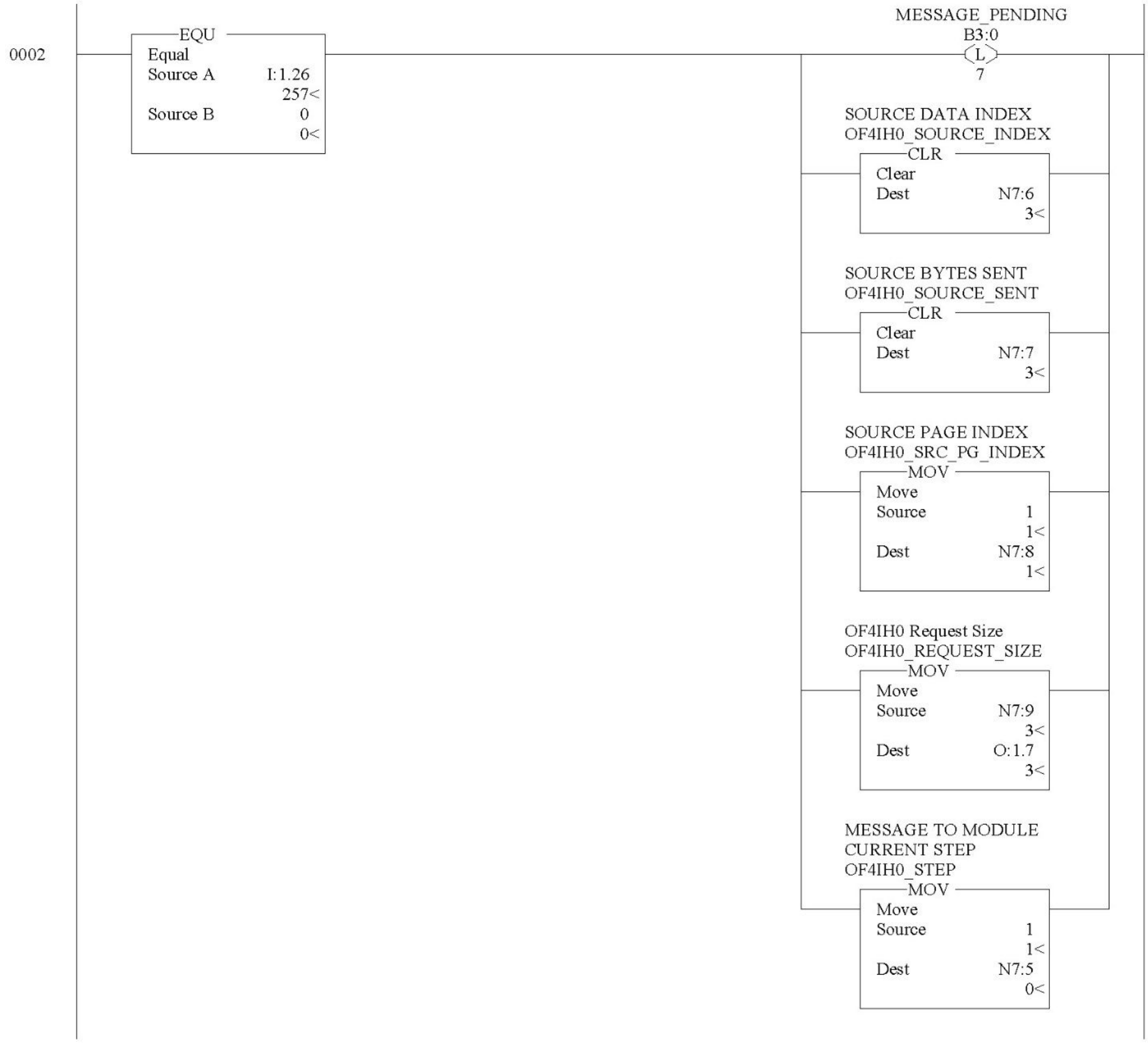

0002
EQU
Equal
Source A I:1.26
257<
Source B 0
0<
MESSAGE_PENDING
B3:0
L
7
SOURCE DATA INDEX
OF4IH0_SOURCE_INDEX
CLR
Clear
Dest N7:6
3<
SOURCE BYTES SENT
OF4IH0_SOURCE_SENT
CLR
Clear
Dest N7:7
3<
SOURCE PAGE INDEX
OF4IH0_SRC_PG_INDEX
MOV
Move
Source 1
1<
Dest N7:8
1<
OF4IH0 Request Size
OF4IH0_REQUEST_SIZE
MOV
Move
Source N7:9
3<
Dest O:1.7
3<
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
MOV
Move
Source 1
1<
Dest N7:5
0<

Figure 7-7c.

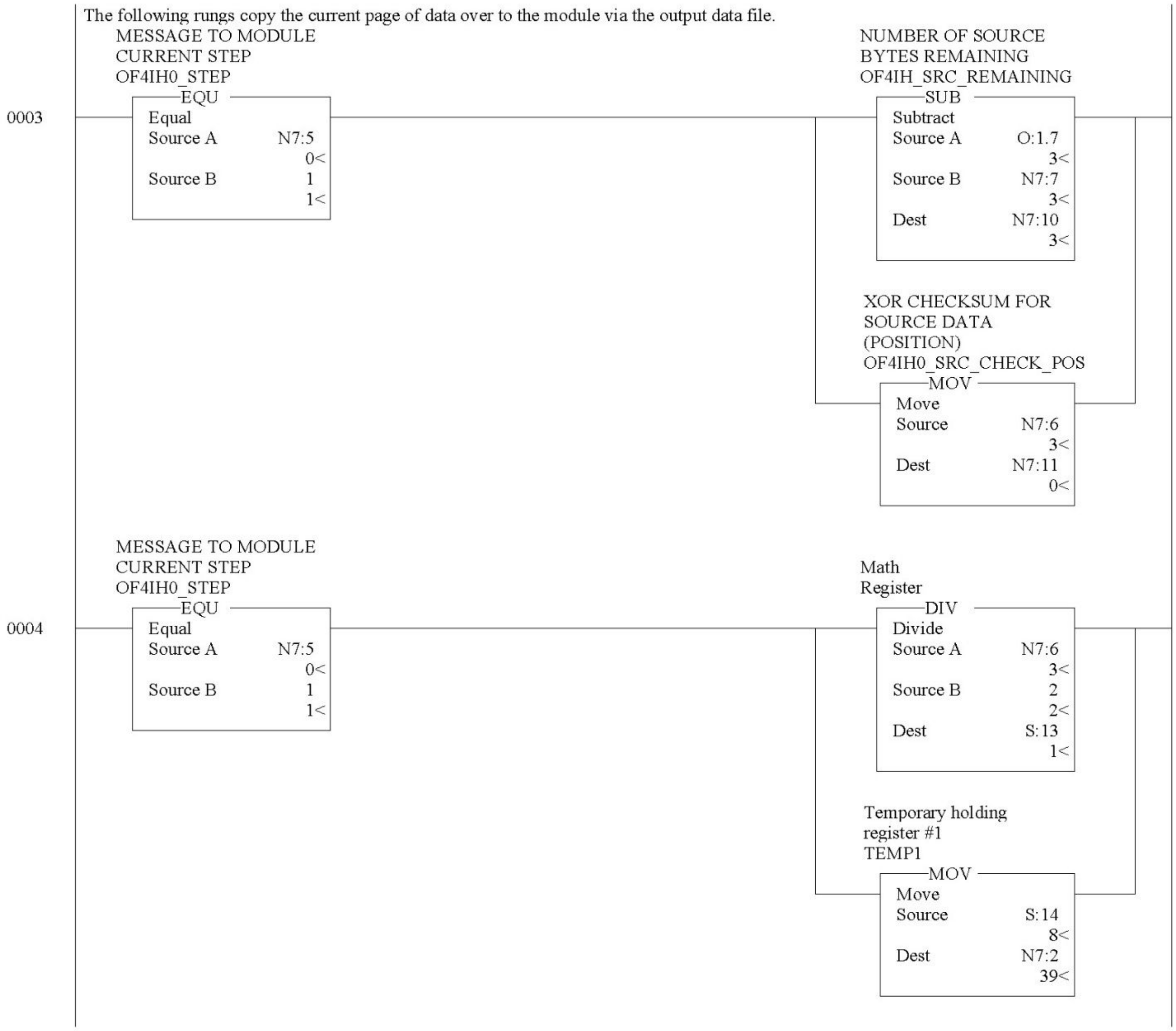

The following rungs copy the current page of data over to the module via the output data file.
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 1
1<
0003
NUMBER OF SOURCE
BYTES REMAINING
OF4IH_SRC_REMAINING
SUB
Subtract
Source A O:1.7
3<
Source B N7:7
3<
Dest N7:10
3<
XOR CHECKSUM FOR
SOURCE DATA
(POSITION)
OF4IH0_SRC_CHECK_POS
MOV
Move
Source N7:6
3<
Dest N7:11
0<
0004
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 1
1<
Math
Register
DIV
Divide
Source A N7:6
3<
Source B 2
2<
Dest S:13
1<
Temporary holding
register #1
TEMP1
MOV
Move
Source S:14
8<
Dest N7:2
39<

Figure 7-7d.

If the bytes remaining are less than 38, this rung copies the data for the current page being sent, to the message request buffer. This rung also calulates the data index.

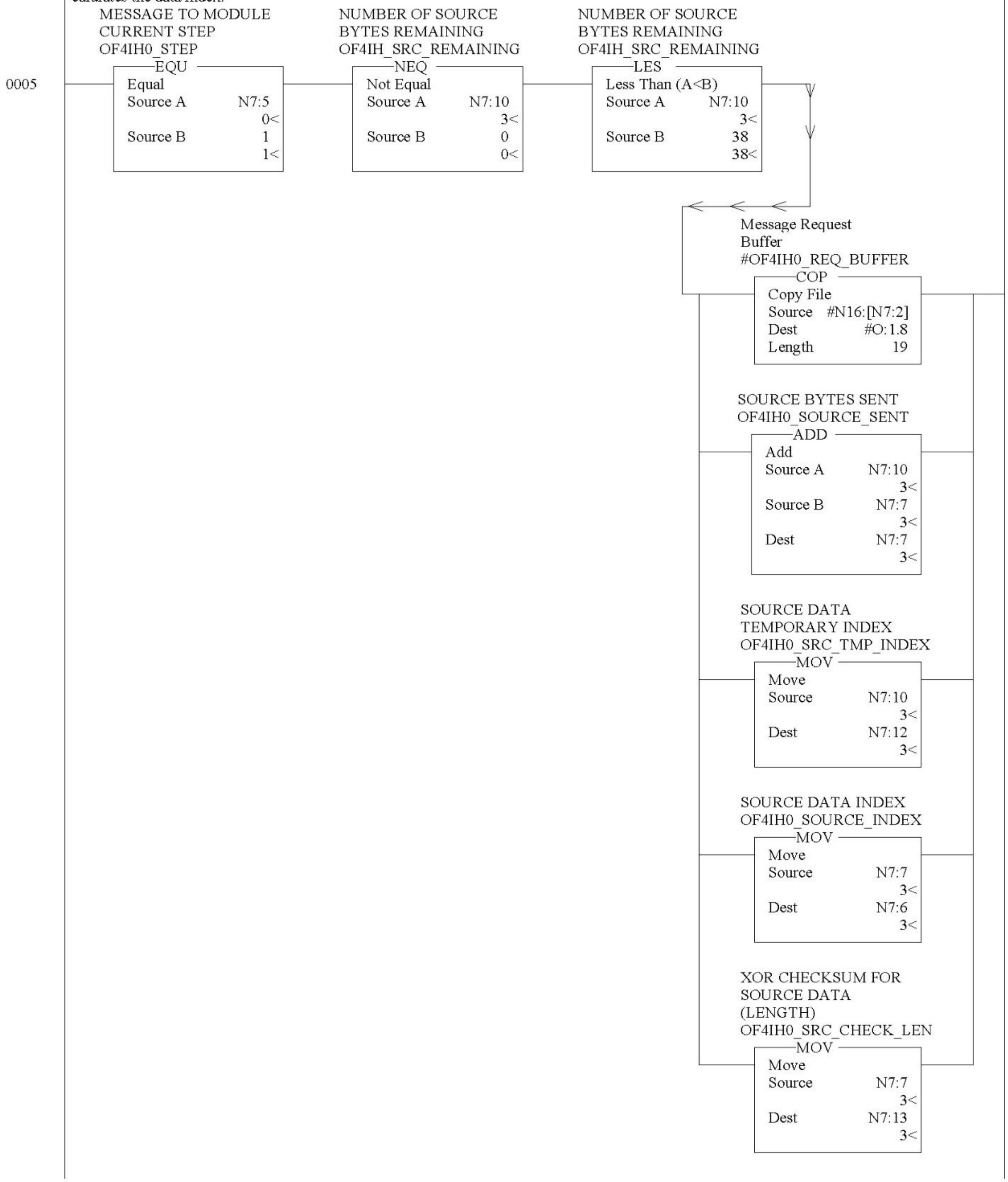


Figure 7-7e.

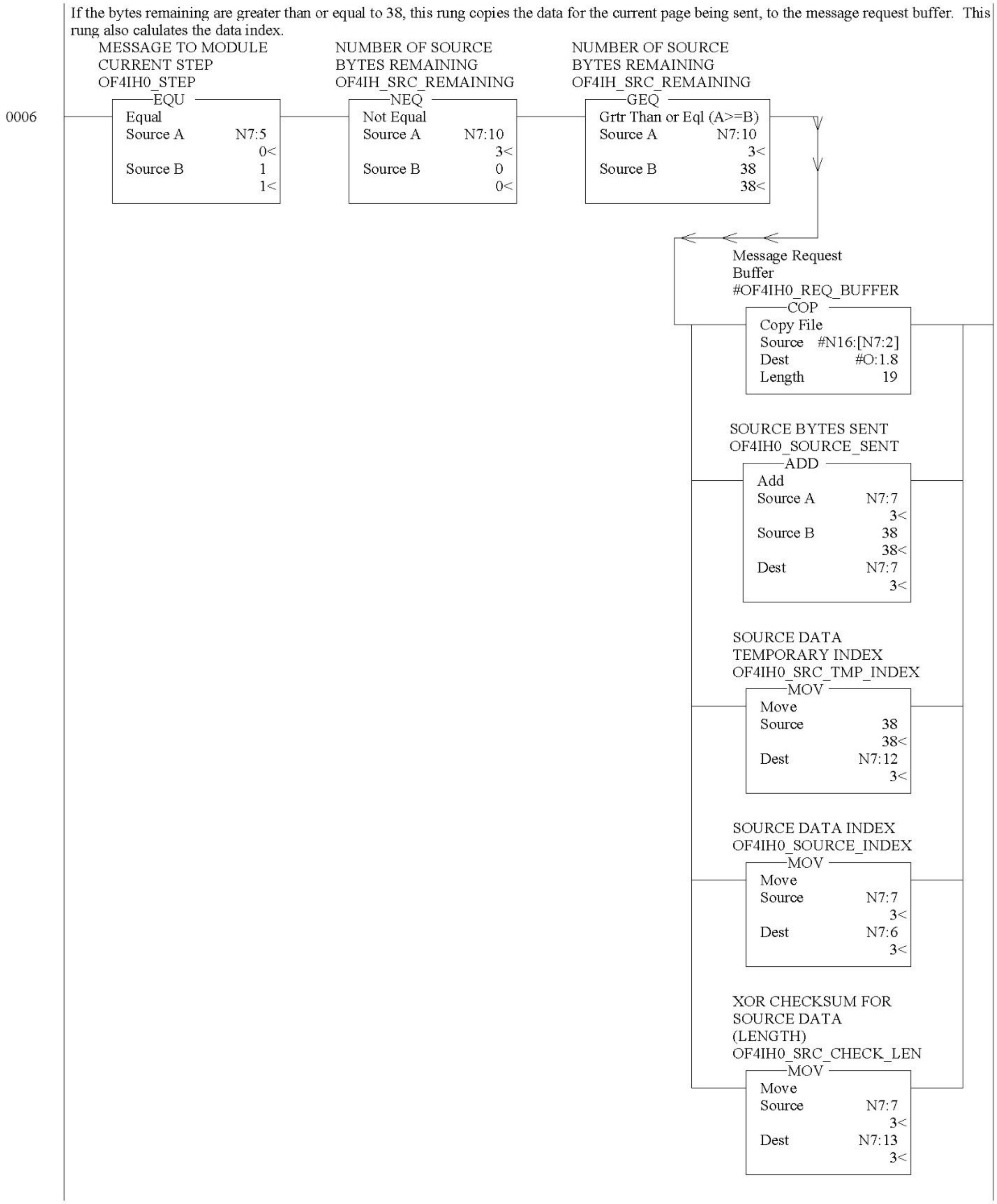

If the bytes remaining are greater than or equal to 38, this rung copies the data for the current page being sent, to the message request buffer. This rung also calulates the data index.
0006
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 1
1<
NUMBER OF SOURCE
BYTES REMAINING
OF4IH_SRC_REMAINING
NEQ
Not Equal
Source A N7:10
3<
Source B 0
0<
NUMBER OF SOURCE
BYTES REMAINING
OF4IH_SRC_REMAINING
GEQ
Grtr Than or Eql (A>=B)
Source A N7:10
3<
Source B 38
38<
Message Request
Buffer
#OF4IH0_REQ_BUFFER
COP
Copy File
Source #N16:[N7:2]
Dest #O:1.8
Length 19
SOURCE BYTES SENT
OF4IH0_SOURCE_SENT
ADD
Add
Source A N7:7
3<
Source B 38
38<
Dest N7:7
3<
SOURCE DATA
TEMPORARY INDEX
OF4IH0_SRC_TMP_INDEX
MOV
Move
Source 38
38<
Dest N7:12
3<
SOURCE DATA INDEX
OF4IH0_SOURCE_INDEX
MOV
Move
Source N7:7
3<
Dest N7:6
3<
XOR CHECKSUM FOR
SOURCE DATA
(LENGTH)
OF4IH0_SRC_CHECK_LEN
MOV
Move
Source N7:7
3<
Dest N7:13
3<

Figure 7-7f.

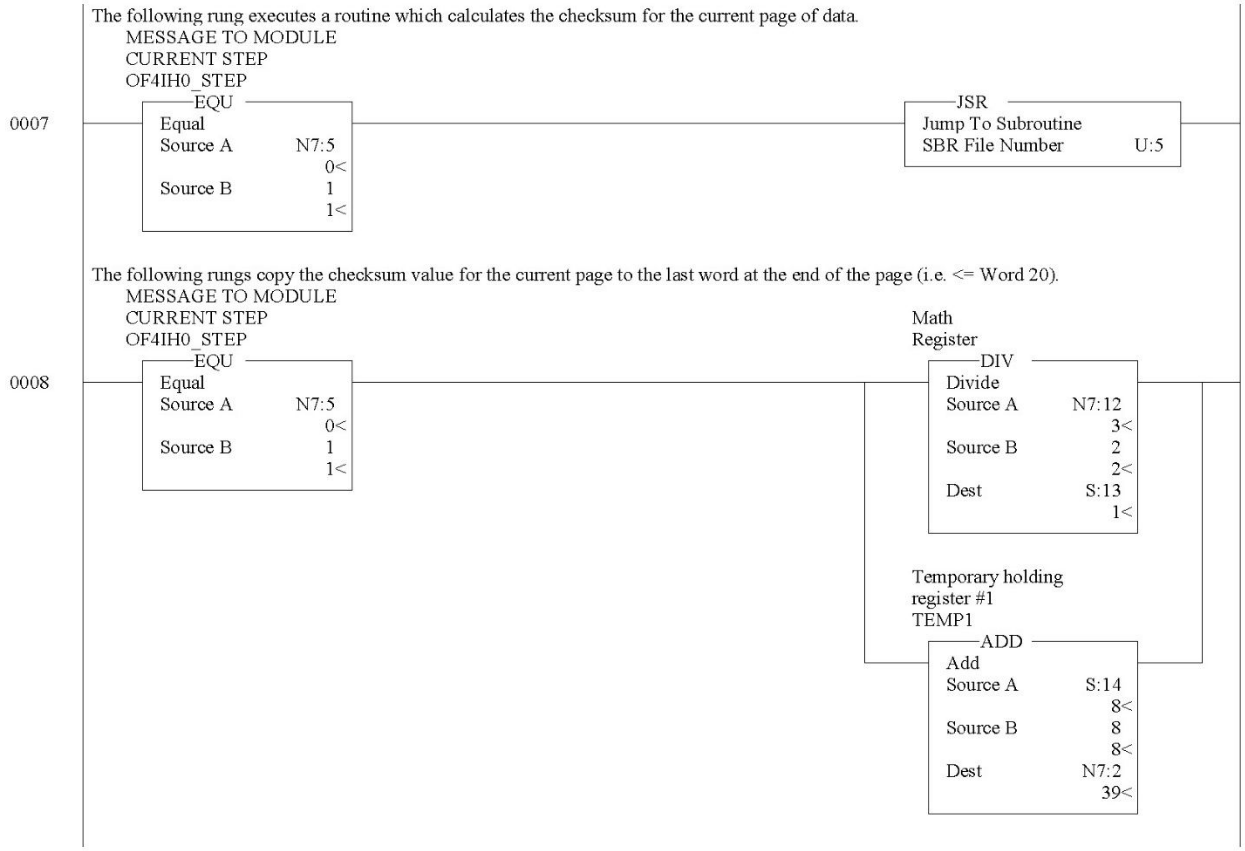

The following rung executes a routine which calculates the checksum for the current page of data.
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
0007
EQU
Equal
Source A N7:5
0<
Source B 1
1<
JSR
Jump To Subroutine
SBR File Number U:5
The following rungs copy the checksum value for the current page to the last word at the end of the page (i.e. <= Word 20).
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
0008
EQU
Equal
Source A N7:5
0<
Source B 1
1<
Math
Register
DIV
Divide
Source A N7:12
3<
Source B 2
2<
Dest S:13
1<
Temporary holding
register #1
TEMP1
ADD
Add
Source A S:14
8<
Source B 8
8<
Dest N7:2
39<

Figure 7-7g.

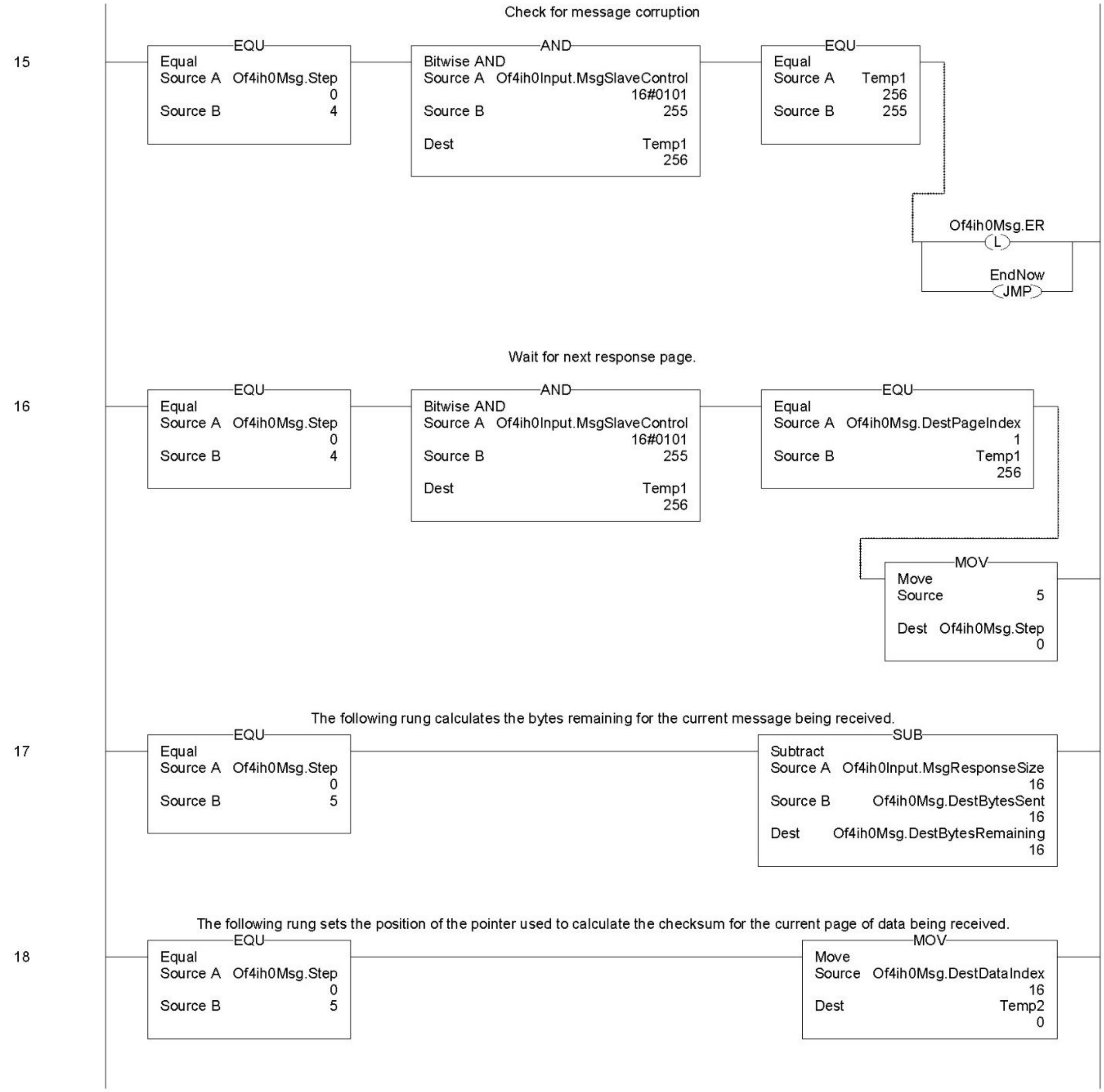

Check for message corruption
15
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 4
AND
Bitwise AND
Source A Of4ih0Input.MsgSlaveControl
16#0101
Source B 255
Dest Temp1
256
EQU
Equal
Source A Temp1
256
Source B 255
Of4ih0Msg.ER
L
EndNow
JMP
Wait for next response page.
16
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 4
AND
Bitwise AND
Source A Of4ih0Input.MsgSlaveControl
16#0101
Source B 255
Dest Temp1
256
EQU
Equal
Source A Of4ih0Msg.DestPageIndex
1
Source B Temp1
256
MOV
Move
Source 5
Dest Of4ih0Msg.Step
0
The following rung calculates the bytes remaining for the current message being received.
17
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 5
SUB
Subtract
Source A Of4ih0Input.MsgResponseSize
16
Source B Of4ih0Msg.DestBytesSent
16
Dest Of4ih0Msg.DestBytesRemaining
16
The following rung sets the position of the pointer used to calculate the checksum for the current page of data being received.
18
EQU
Equal
Source A Of4ih0Msg.Step
0
Source B 5
MOV
Move
Source Of4ih0Msg.DestDataIndex
16
Dest Temp2
0

Figure 7-7h.

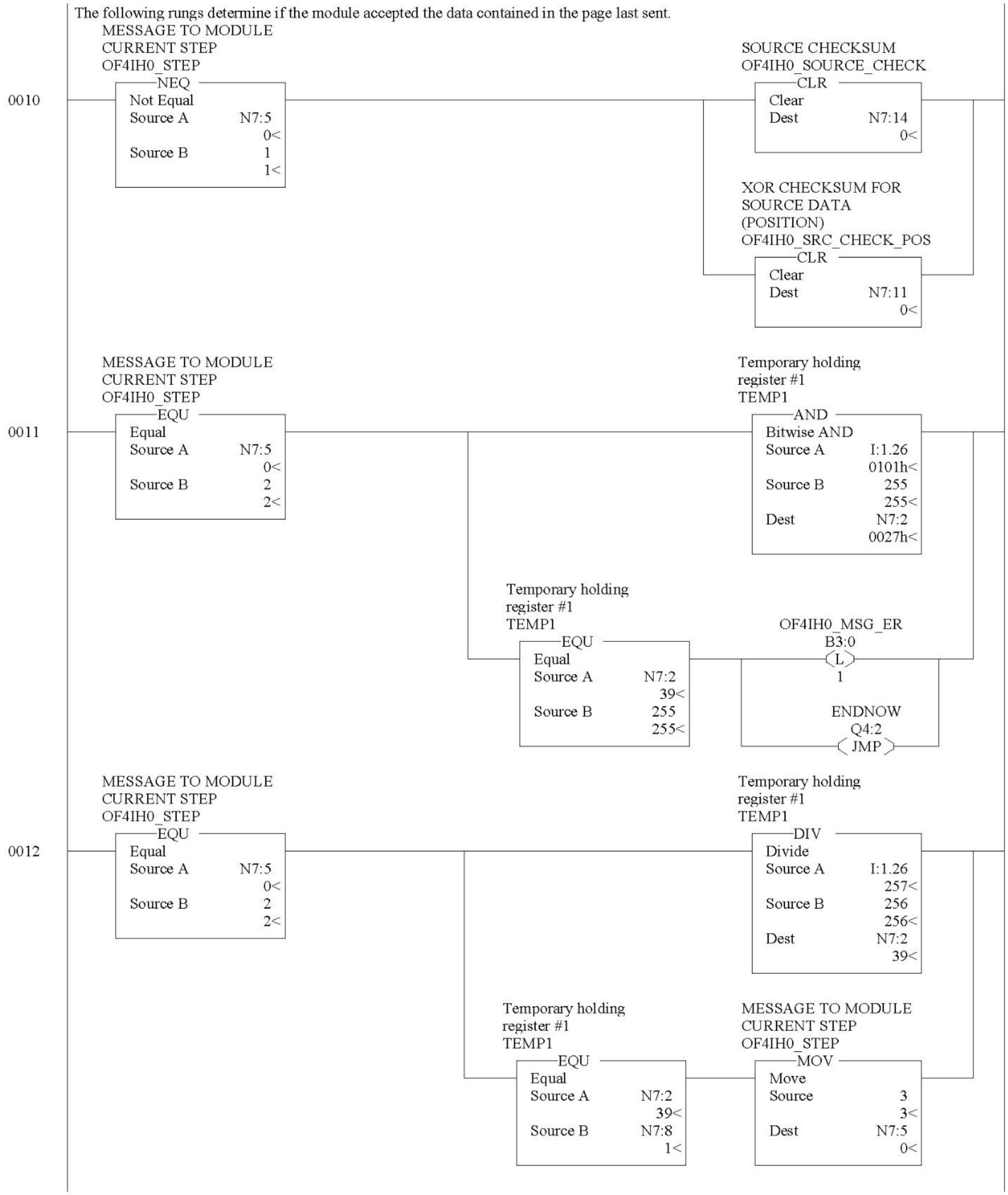

The following rungs determine if the module accepted the data contained in the page last sent.
0010
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
NEQ
Not Equal
Source A N7:5
0<
Source B 1
1<
SOURCE CHECKSUM
OF4IH0_SOURCE_CHECK
CLR
Clear
Dest N7:14
0<
XOR CHECKSUM FOR
SOURCE DATA
(POSITION)
OF4IH0_SRC_CHECK_POS
CLR
Clear
Dest N7:11
0<
0011
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 2
2<
Temporary holding
register #1
TEMP1
AND
Bitwise AND
Source A I:1.26
0101h<
Source B 255
255<
Dest N7:2
0027h<
Temporary holding
register #1
TEMP1
EQU
Equal
Source A N7:2
39<
Source B 255
255<
OF4IH0_MSG_ER
B3:0
L
1
ENDNOW
Q4:2
JMP
0012
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 2
2<
Temporary holding
register #1
TEMP1
DIV
Divide
Source A I:1.26
257<
Source B 256
256<
Dest N7:2
39<
Temporary holding
register #1
TEMP1
EQU
Equal
Source A N7:2
39<
Source B N7:8
1<
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
MOV
Move
Source 3
3<
Dest N7:5
0<

Figure 7-7i.

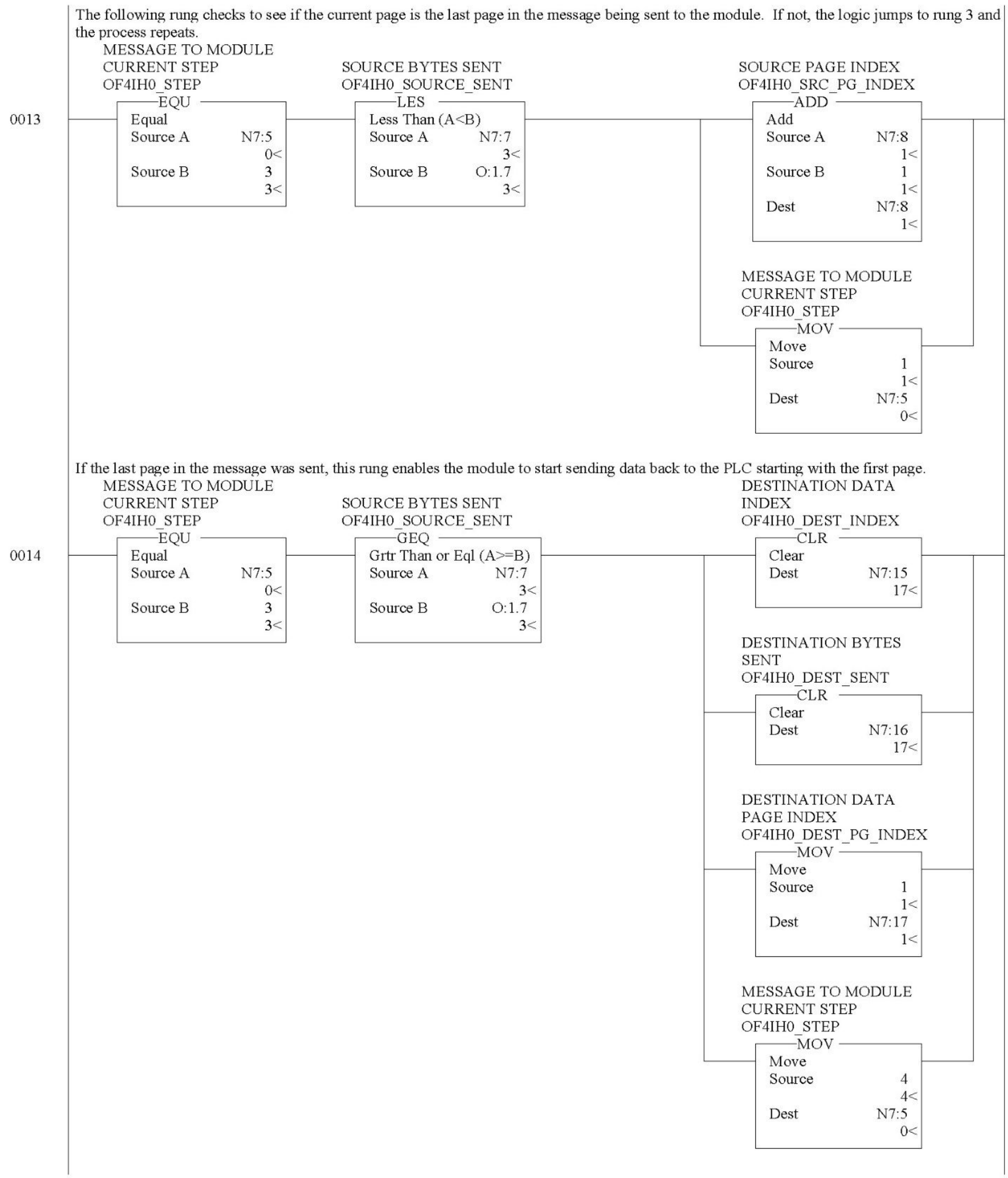

The following rung checks to see if the current page is the last page in the message being sent to the module. If not, the logic jumps to rung 3 and the process repeats.
0013
MESSAGE TO MODULE CURRENT STEP OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 3
3<
SOURCE BYTES SENT OF4IH0_SOURCE_SENT
LES
Less Than (A<B)
Source A N7:7
3<
Source B O:1.7
3<
SOURCE PAGE INDEX OF4IH0_SRC_PG_INDEX
ADD
Add
Source A N7:8
1<
Source B 1
1<
Dest N7:8
1<
MESSAGE TO MODULE CURRENT STEP OF4IH0_STEP
MOV
Move
Source 1
1<
Dest N7:5
0<
If the last page in the message was sent, this rung enables the module to start sending data back to the PLC starting with the first page.
0014
MESSAGE TO MODULE CURRENT STEP OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 3
3<
SOURCE BYTES SENT OF4IH0_SOURCE_SENT
GEQ
Grtr Than or Eql (A>=B)
Source A N7:7
3<
Source B O:1.7
3<
DESTINATION DATA INDEX OF4IH0_DEST_INDEX
CLR
Clear
Dest N7:15
17<
DESTINATION BYTES SENT OF4IH0_DEST_SENT
CLR
Clear
Dest N7:16
17<
DESTINATION DATA PAGE INDEX OF4IH0_DEST_PG_INDEX
MOV
Move
Source 1
1<
Dest N7:17
1<
MESSAGE TO MODULE CURRENT STEP OF4IH0_STEP
MOV
Move
Source 4
4<
Dest N7:5
0<

Figure 7-7j.

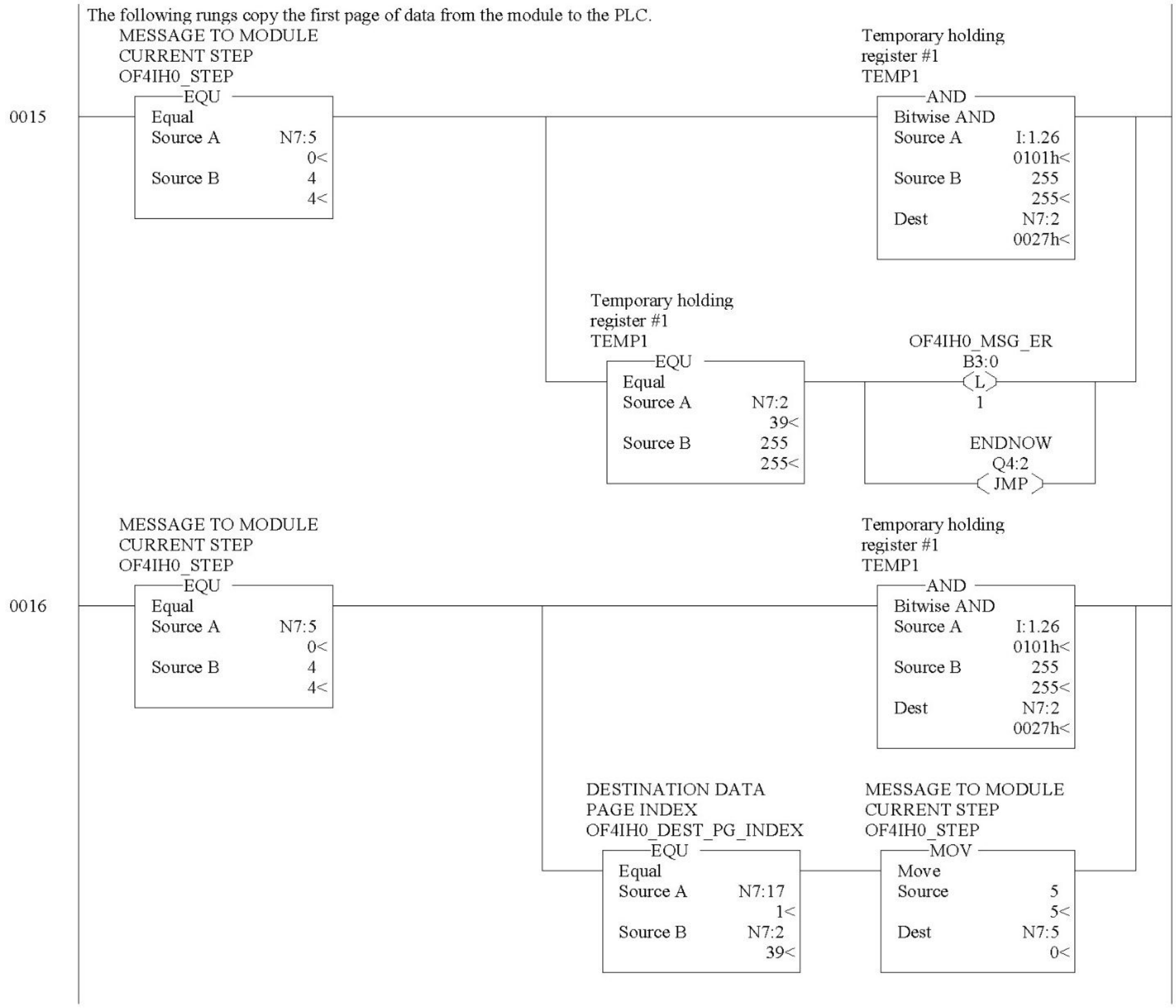
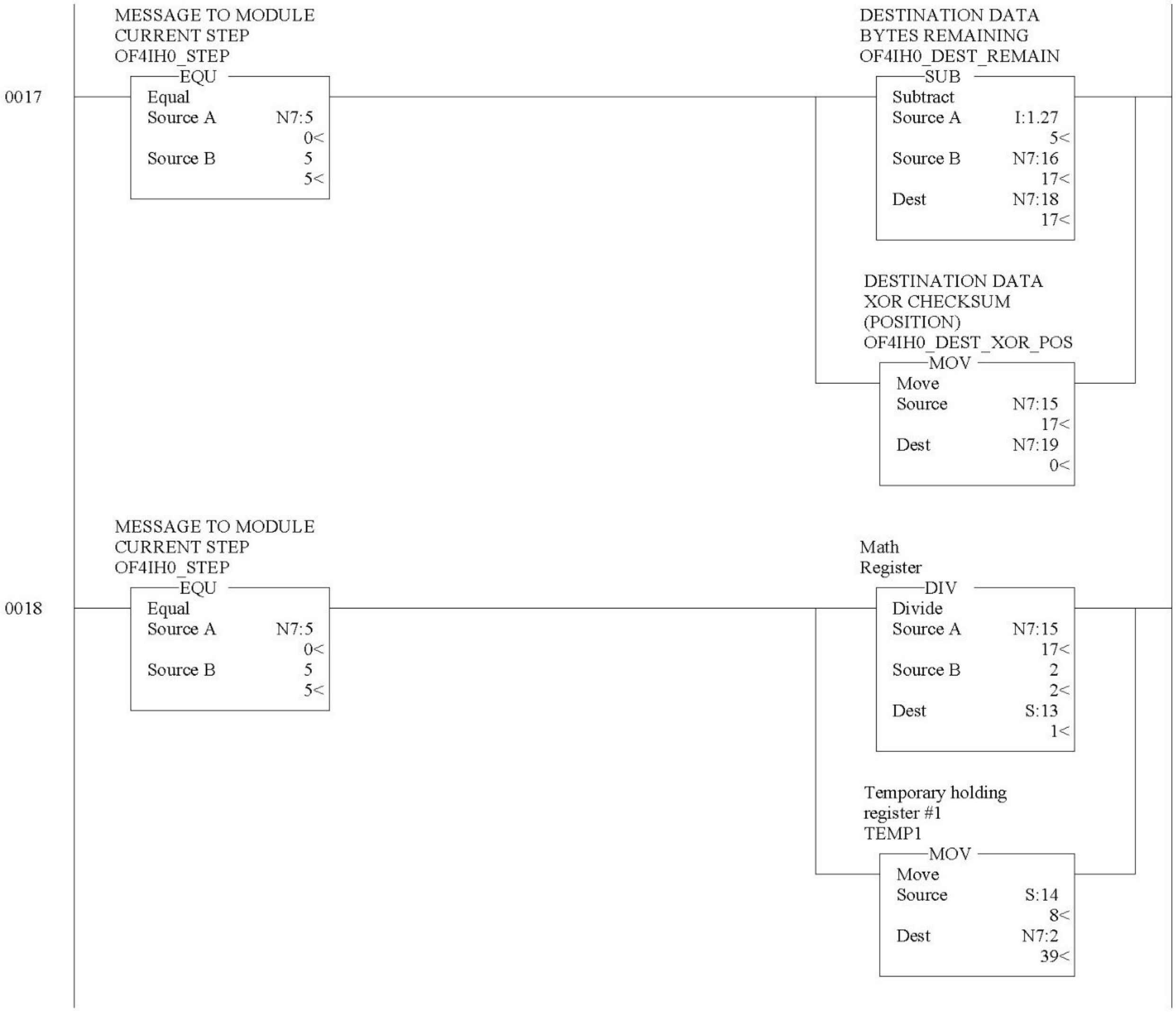

The following rungs copy the first page of data from the module to the PLC.
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
0015
EQU
Equal
Source A N7:5
0<
Source B 4
4<
Temporary holding
register #1
TEMP1
AND
Bitwise AND
Source A I:1.26
0101h<
Source B 255
255<
Dest N7:2
0027h<
Temporary holding
register #1
TEMP1
EQU
Equal
Source A N7:2
39<
Source B 255
255<
OF4IH0_MSG_ER
B3:0
L
1
ENDNOW
Q4:2
JMP
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
0016
EQU
Equal
Source A N7:5
0<
Source B 4
4<
Temporary holding
register #1
TEMP1
AND
Bitwise AND
Source A I:1.26
0101h<
Source B 255
255<
Dest N7:2
0027h<
DESTINATION DATA
PAGE INDEX
OF4IH0_DEST_PG_INDEX
EQU
Equal
Source A N7:17
1<
Source B N7:2
39<
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
MOV
Move
Source 5
5<
Dest N7:5
0<

Figure 7-7k.



MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 5
5<
0017
DESTINATION DATA
BYTES REMAINING
OF4IH0_DEST_REMAIN
SUB
Subtract
Source A I:1.27
5<
Source B N7:16
17<
Dest N7:18
17<
DESTINATION DATA
XOR CHECKSUM
(POSITION)
OF4IH0_DEST_XOR_POS
MOV
Move
Source N7:15
17<
Dest N7:19
0<
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 5
5<
0018
Math
Register
DIV
Divide
Source A N7:15
17<
Source B 2
2<
Dest S:13
1<
Temporary holding
register #1
TEMP1
MOV
Move
Source S:14
8<
Dest N7:2
39<

Figure 7-7l.

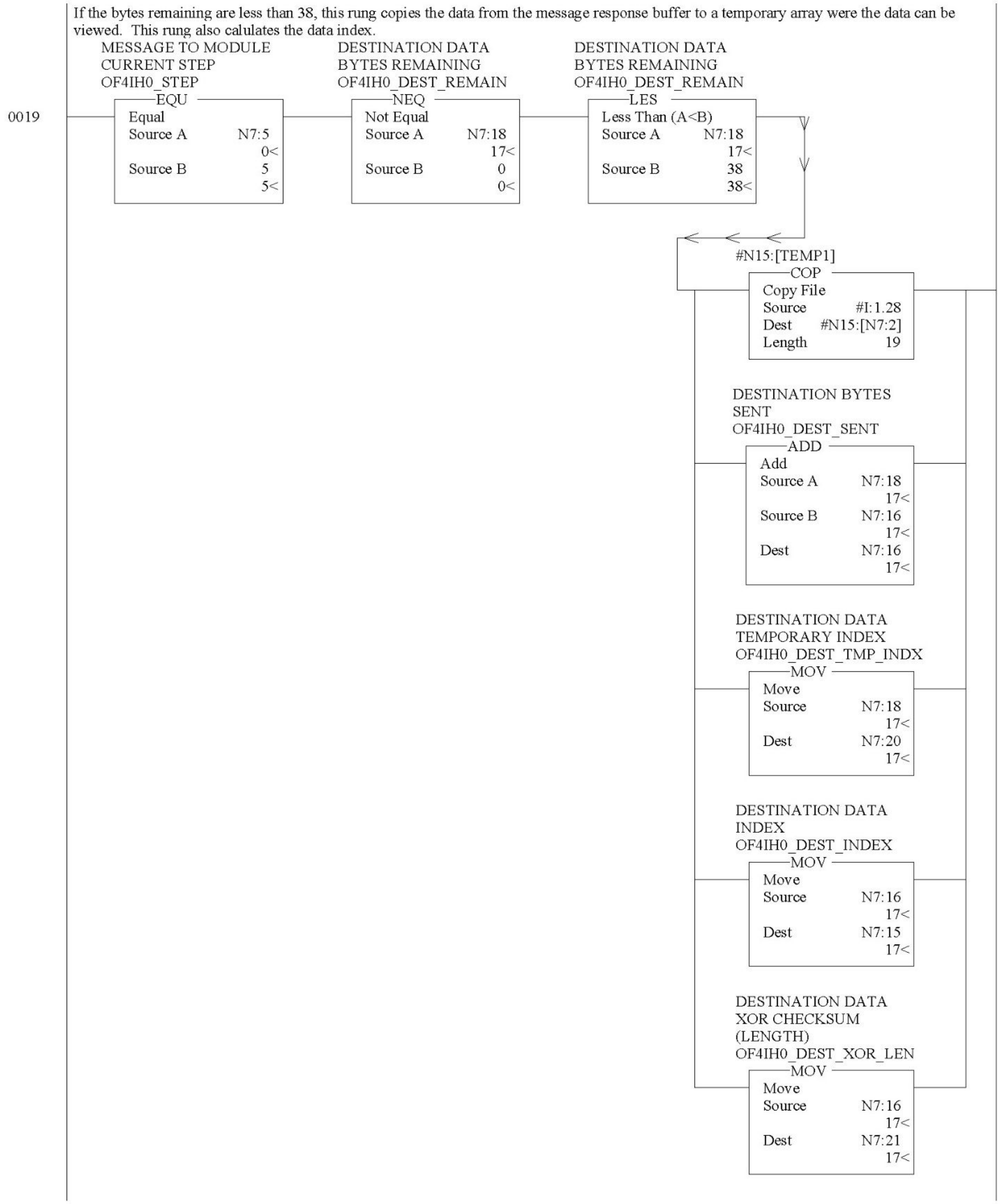

If the bytes remaining are less than 38, this rung copies the data from the message response buffer to a temporary array were the data can be viewed. This rung also calulates the data index.
0019
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 5
5<
DESTINATION DATA
BYTES REMAINING
OF4IH0_DEST_REMAIN
NEQ
Not Equal
Source A N7:18
17<
Source B 0
0<
DESTINATION DATA
BYTES REMAINING
OF4IH0_DEST_REMAIN
LES
Less Than (A<B)
Source A N7:18
17<
Source B 38
38<
#N15:[TEMP1]
COP
Copy File
Source #I:1.28
Dest #N15:[N7:2]
Length 19
DESTINATION BYTES
SENT
OF4IH0_DEST_SENT
ADD
Add
Source A N7:18
17<
Source B N7:16
17<
Dest N7:16
17<
DESTINATION DATA
TEMPORARY INDEX
OF4IH0_DEST_TMP_INDX
MOV
Move
Source N7:18
17<
Dest N7:20
17<
DESTINATION DATA
INDEX
OF4IH0_DEST_INDEX
MOV
Move
Source N7:16
17<
Dest N7:15
17<
DESTINATION DATA
XOR CHECKSUM
(LENGTH)
OF4IH0_DEST_XOR_LEN
MOV
Move
Source N7:16
17<
Dest N7:21
17<

Figure 7-7m.

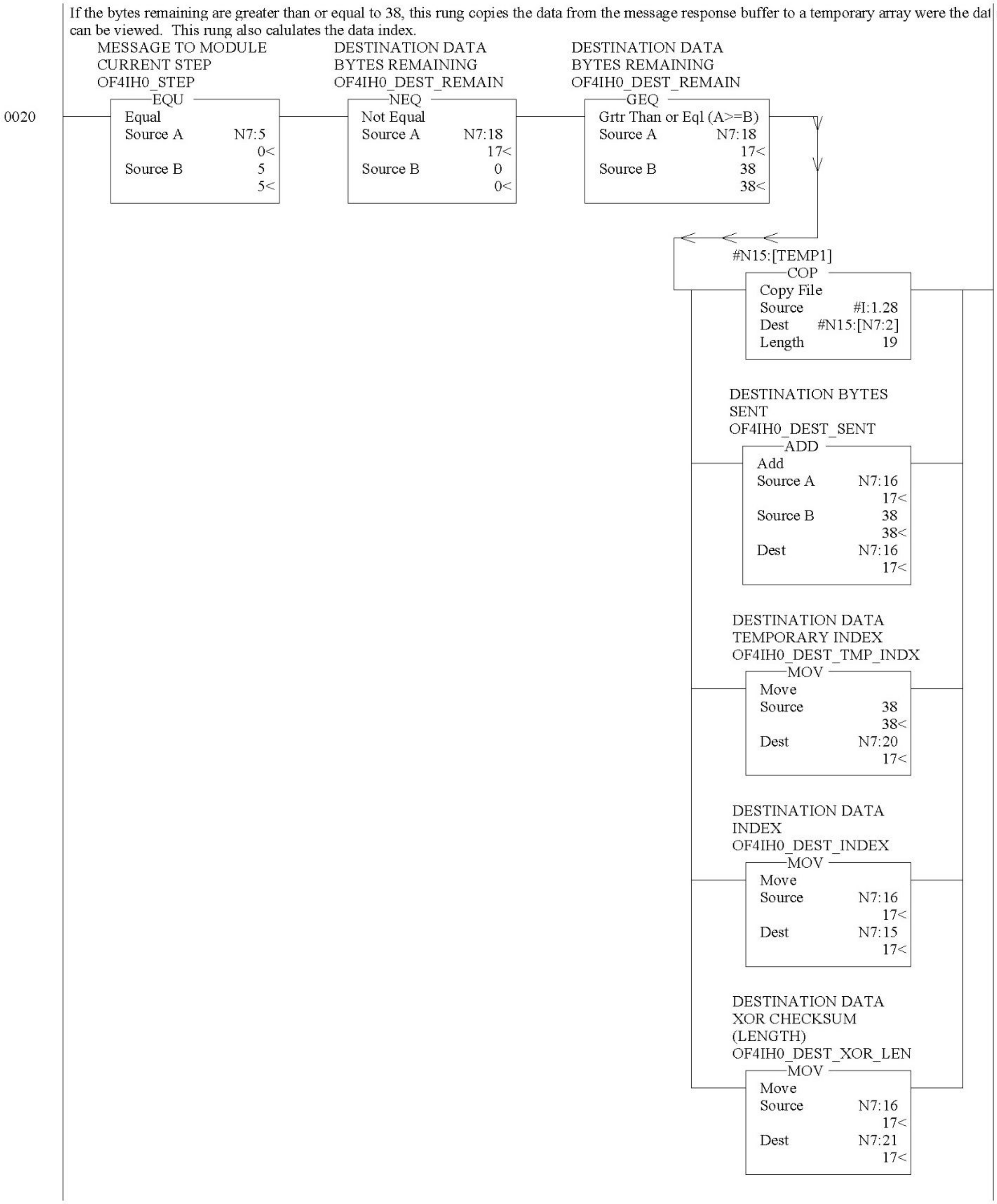

If the bytes remaining are greater than or equal to 38, this rung copies the data from the message response buffer to a temporary array were the dat
can be viewed. This rung also calulates the data index.
0020
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 5
5<
DESTINATION DATA
BYTES REMAINING
OF4IH0_DEST_REMAIN
NEQ
Not Equal
Source A N7:18
17<
Source B 0
0<
DESTINATION DATA
BYTES REMAINING
OF4IH0_DEST_REMAIN
GEQ
Grtr Than or Eql (A>=B)
Source A N7:18
17<
Source B 38
38<
#N15:[TEMP1]
COP
Copy File
Source #I:1.28
Dest #N15:[N7:2]
Length 19
DESTINATION BYTES
SENT
OF4IH0_DEST_SENT
ADD
Add
Source A N7:16
17<
Source B 38
38<
Dest N7:16
17<
DESTINATION DATA
TEMPORARY INDEX
OF4IH0_DEST_TMP_INDX
MOV
Move
Source 38
38<
Dest N7:20
17<
DESTINATION DATA
INDEX
OF4IH0_DEST_INDEX
MOV
Move
Source N7:16
17<
Dest N7:15
17<
DESTINATION DATA
XOR CHECKSUM
(LENGTH)
OF4IH0_DEST_XOR_LEN
MOV
Move
Source N7:16
17<
Dest N7:21
17<

Figure 7-7n.

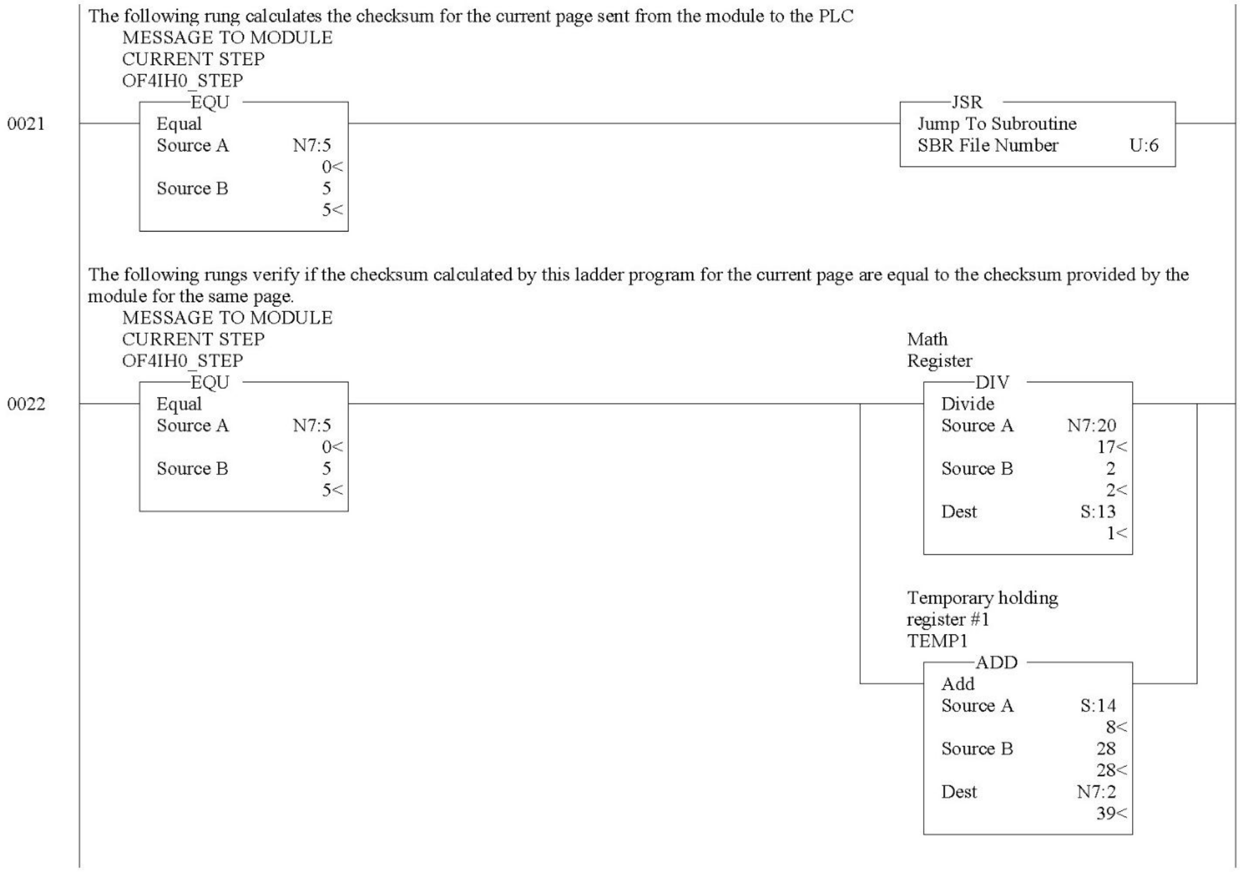

The following rung calculates the checksum for the current page sent from the module to the PLC
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
0021
EQU
Equal
Source A N7:5
0<
Source B 5
5<
JSR
Jump To Subroutine
SBR File Number U:6
The following rungs verify if the checksum calculated by this ladder program for the current page are equal to the checksum provided by the module for the same page.
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
0022
EQU
Equal
Source A N7:5
0<
Source B 5
5<
Math
Register
DIV
Divide
Source A N7:20
17<
Source B 2
2<
Dest S:13
1<
Temporary holding
register #1
TEMP1
ADD
Add
Source A S:14
8<
Source B 28
28<
Dest N7:2
39<

Figure 7-7o.

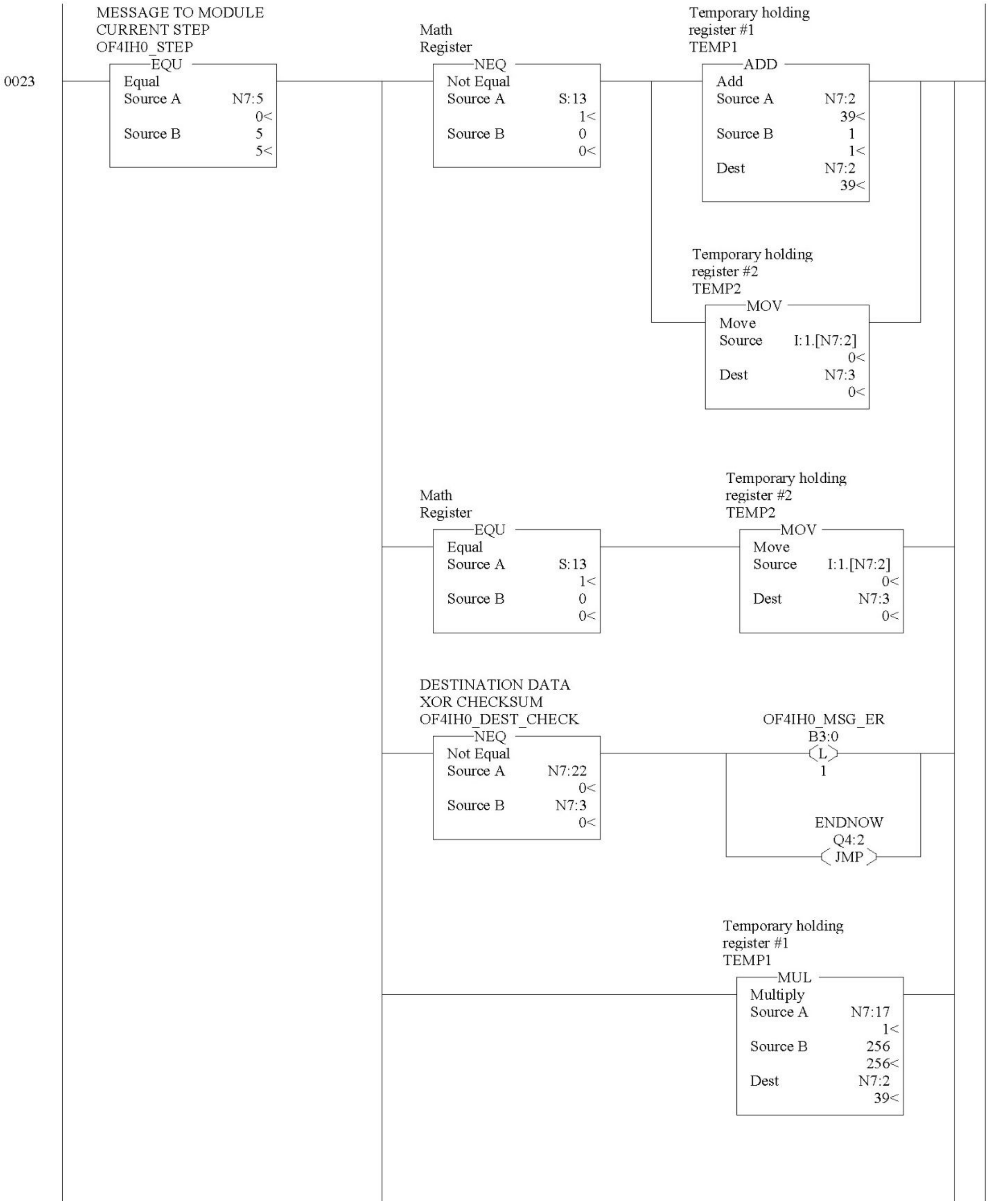

0023
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 5
5<
Math
Register
NEQ
Not Equal
Source A S:13
1<
Source B 0
0<
Temporary holding
register #1
TEMP1
ADD
Add
Source A N7:2
39<
Source B 1
1<
Dest N7:2
39<
Temporary holding
register #2
TEMP2
MOV
Move
Source I:1.[N7:2]
0<
Dest N7:3
0<
Math
Register
EQU
Equal
Source A S:13
1<
Source B 0
0<
Temporary holding
register #2
TEMP2
MOV
Move
Source I:1.[N7:2]
0<
Dest N7:3
0<
DESTINATION DATA
XOR CHECKSUM
OF4IH0_DEST_CHECK
NEQ
Not Equal
Source A N7:22
0<
Source B N7:3
0<
OF4IH0_MSG_ER
B3:0
L
1
ENDNOW
Q4:2
JMP
Temporary holding
register #1
TEMP1
MUL
Multiply
Source A N7:17
1<
Source B 256
256<
Dest N7:2
39<

Figure 7-7p.

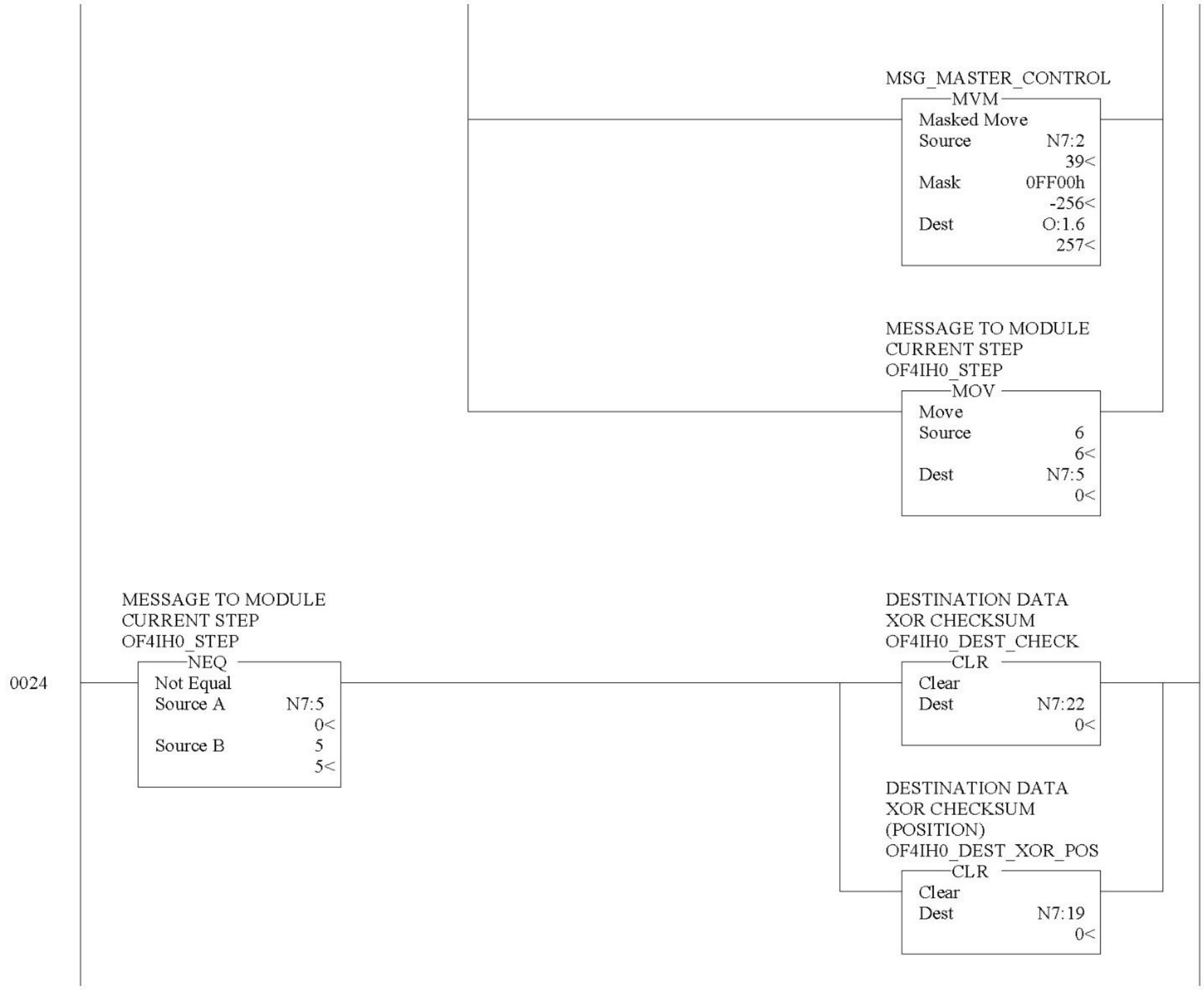
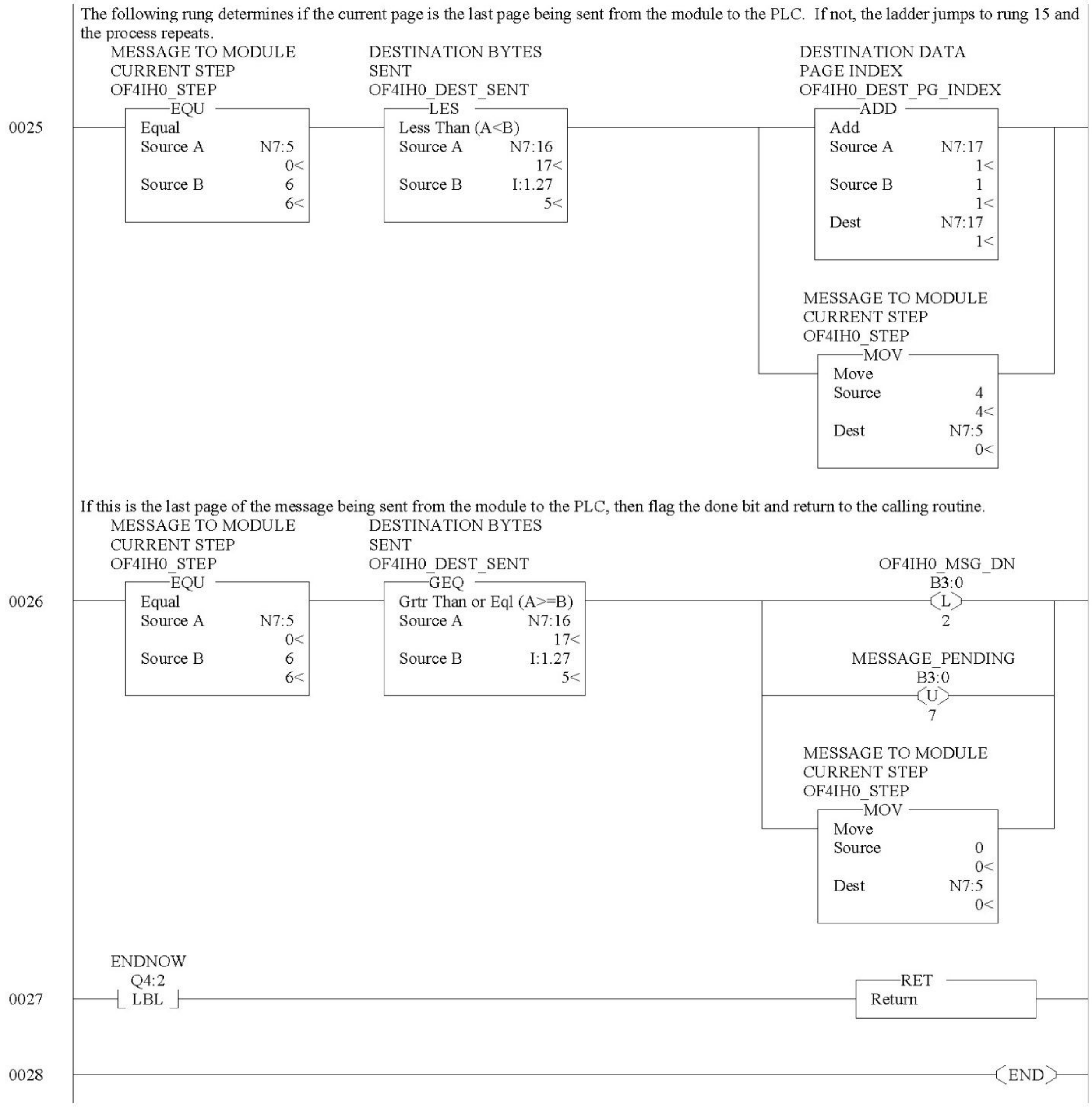

MSG_MASTER_CONTROL
MVM
Masked Move
Source N7:2
39<
Mask 0FF00h
-256<
Dest O:1.6
257<
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
MOV
Move
Source 6
6<
Dest N7:5
0<
0024
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
NEQ
Not Equal
Source A N7:5
0<
Source B 5
5<
DESTINATION DATA
XOR CHECKSUM
OF4IH0_DEST_CHECK
CLR
Clear
Dest N7:22
0<
DESTINATION DATA
XOR CHECKSUM
(POSITION)
OF4IH0_DEST_XOR_POS
CLR
Clear
Dest N7:19
0<

Figure 7-7q.


The following rung determines if the current page is the last page being sent from the module to the PLC. If not, the ladder jumps to rung 15 and the process repeats.
0025
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 6
6<
DESTINATION BYTES
SENT
OF4IH0_DEST_SENT
LES
Less Than (A<B)
Source A N7:16
17<
Source B I:1.27
5<
DESTINATION DATA
PAGE INDEX
OF4IH0_DEST_PG_INDEX
ADD
Add
Source A N7:17
1<
Source B 1
1<
Dest N7:17
1<
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
MOV
Move
Source 4
4<
Dest N7:5
0<
If this is the last page of the message being sent from the module to the PLC, then flag the done bit and return to the calling routine.
0026
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
EQU
Equal
Source A N7:5
0<
Source B 6
6<
DESTINATION BYTES
SENT
OF4IH0_DEST_SENT
GEQ
Grtr Than or Eql (A>=B)
Source A N7:16
17<
Source B I:1.27
5<
OF4IH0_MSG_DN
B3:0
L
2
MESSAGE_PENDING
B3:0
U
7
MESSAGE TO MODULE
CURRENT STEP
OF4IH0_STEP
MOV
Move
Source 0
0<
Dest N7:5
0<
0027
ENDNOW
Q4:2
LBL
RET
Return
0028
END

7.2.4 SRC_CHECK Routine

Calculates the checksum for a message sent to the module one page at a time. This routine is called from the MSG_TO_MOD routine.

Figure 7-8a. Source Checksum

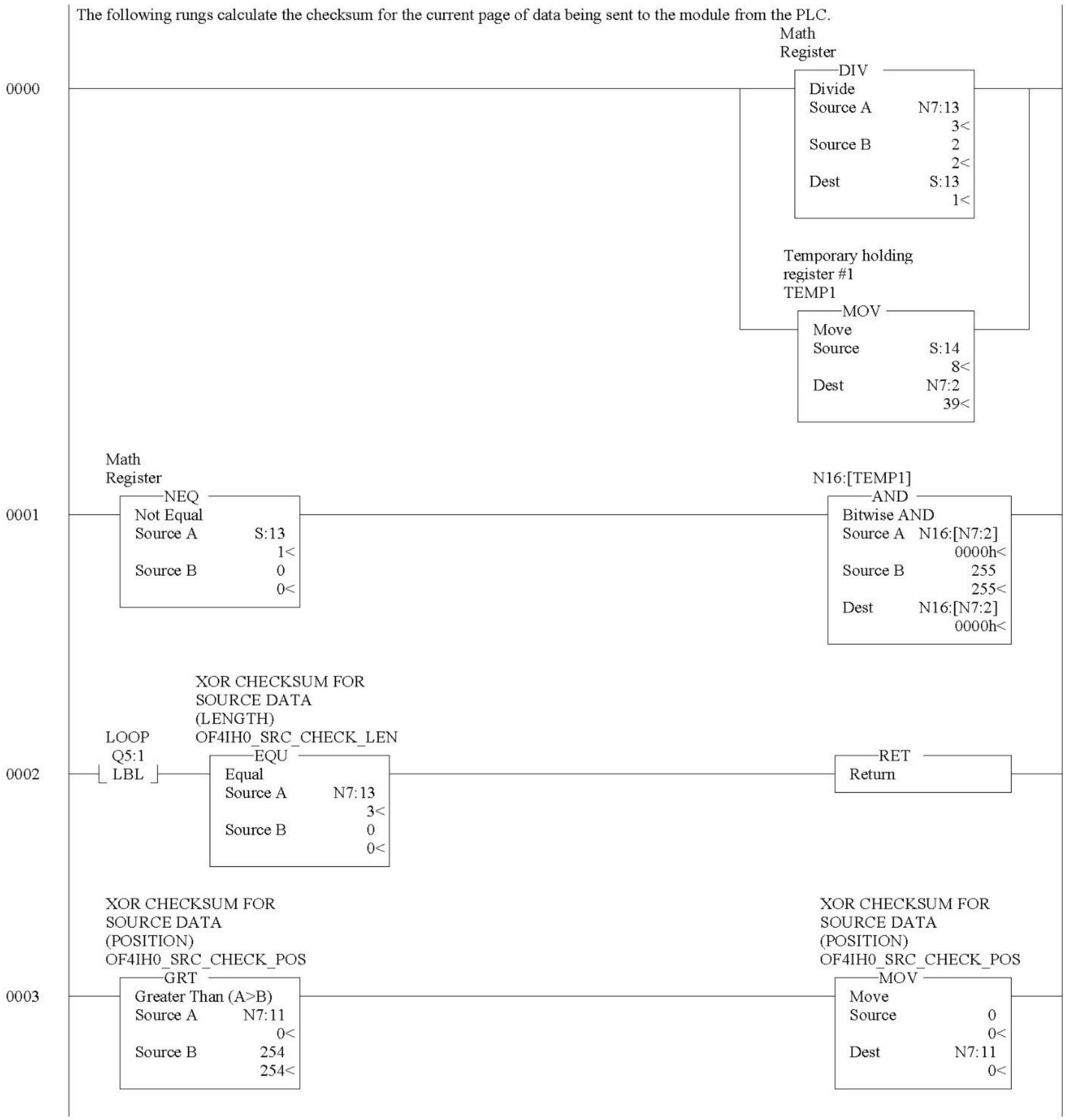
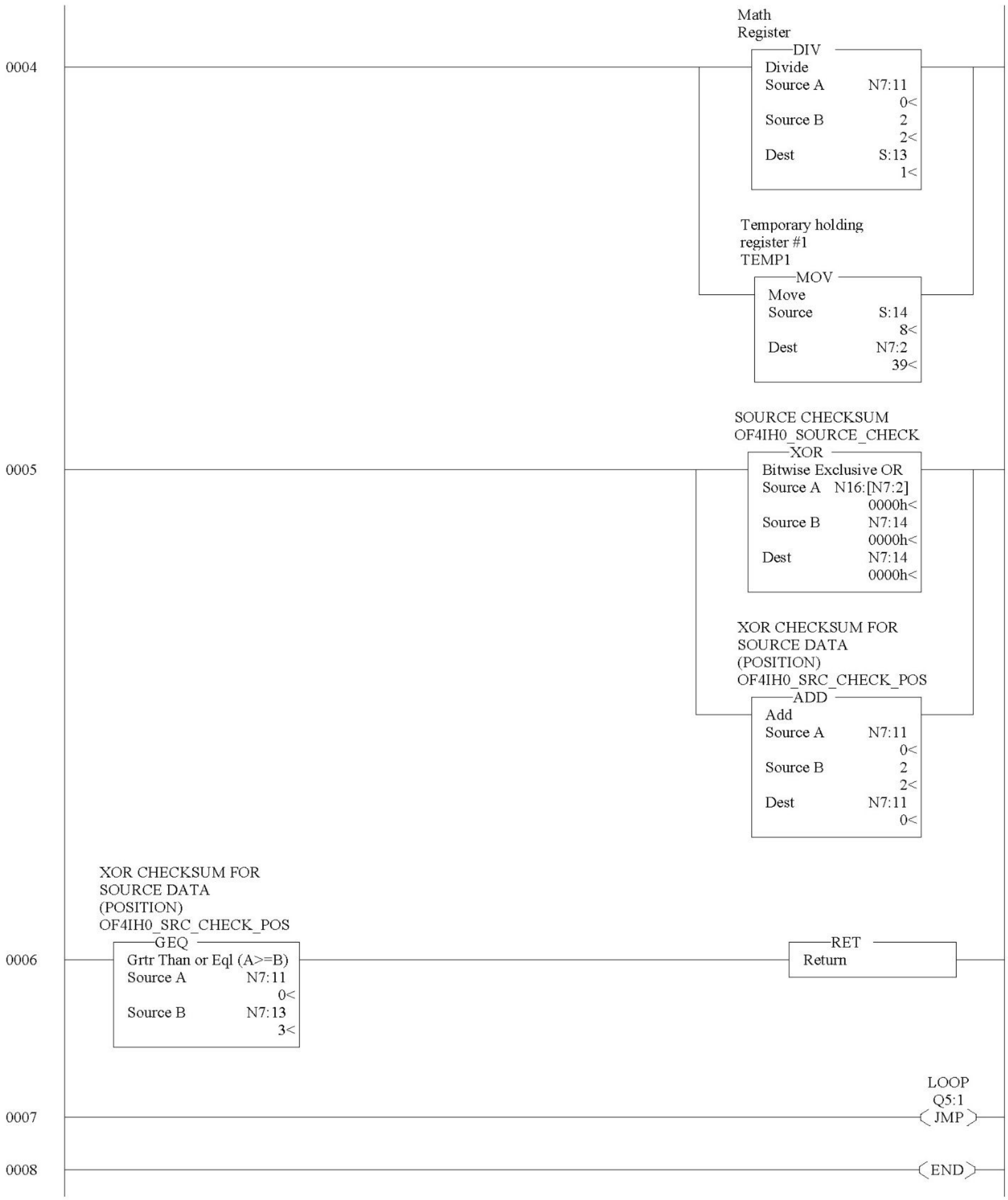


Figure 7-8b.


0004
Math
Register
DIV
Divide
Source A N7:11
0<
Source B 2
2<
Dest S:13
1<
Temporary holding
register #1
TEMP1
MOV
Move
Source S:14
8<
Dest N7:2
39<
0005
SOURCE CHECKSUM
OF4IH0_SOURCE_CHECK
XOR
Bitwise Exclusive OR
Source A N16:[N7:2]
0000h<
Source B N7:14
0000h<
Dest N7:14
0000h<
XOR CHECKSUM FOR
SOURCE DATA
(POSITION)
OF4IH0_SRC_CHECK_POS
ADD
Add
Source A N7:11
0<
Source B 2
2<
Dest N7:11
0<
0006
XOR CHECKSUM FOR
SOURCE DATA
(POSITION)
OF4IH0_SRC_CHECK_POS
GEQ
Grtr Than or Eql (A>=B)
Source A N7:11
0<
Source B N7:13
3<
RET
Return
0007
LOOP
Q5:1
JMP
0008
END

7.2.5 DEST_CHECKSUM Routine

This routine calculates the checksum for a message received from the module one page at a time. This routine is called from the MSG_TO_MOD routine.

Figure 7-9a. Destination Checksum

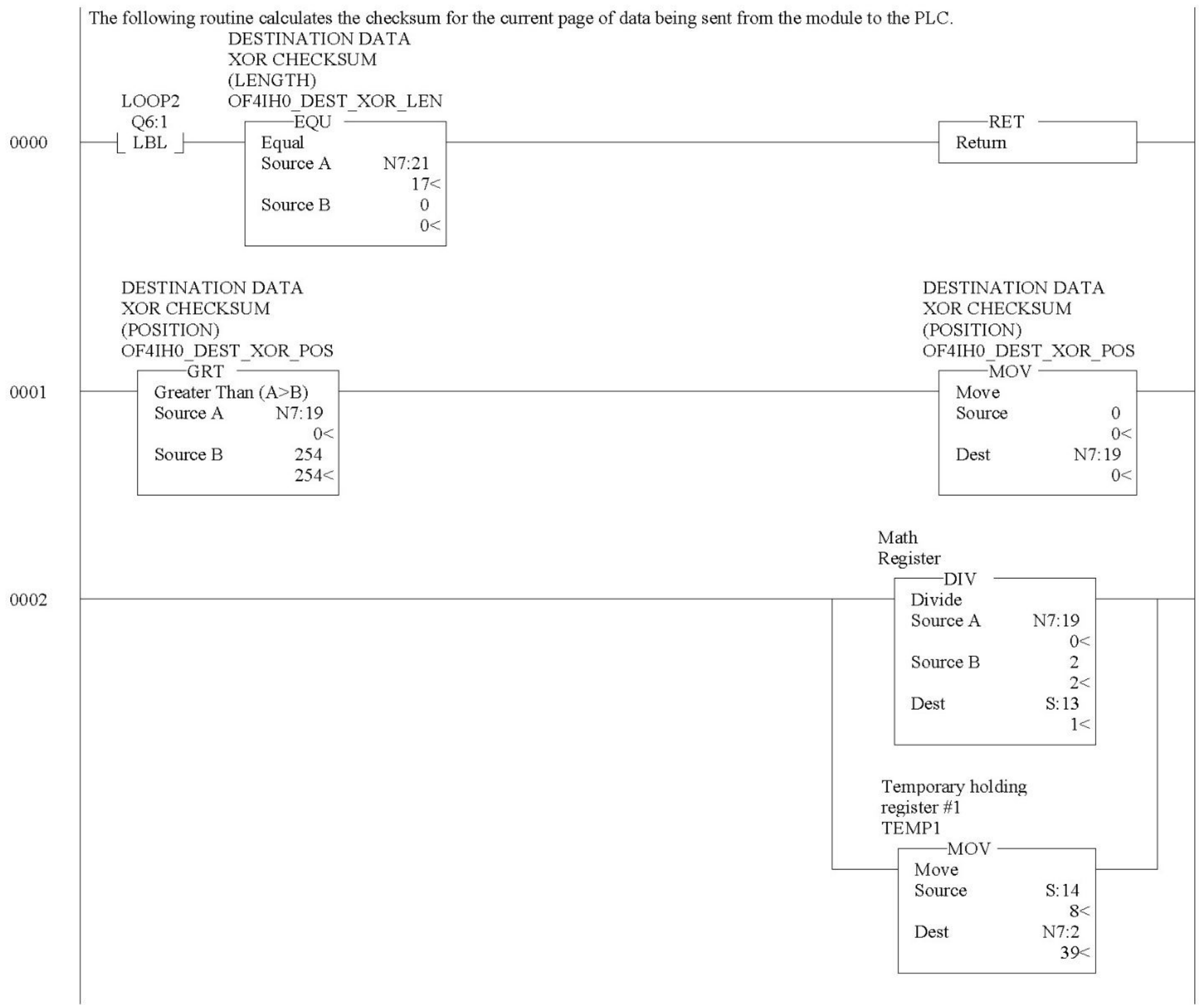
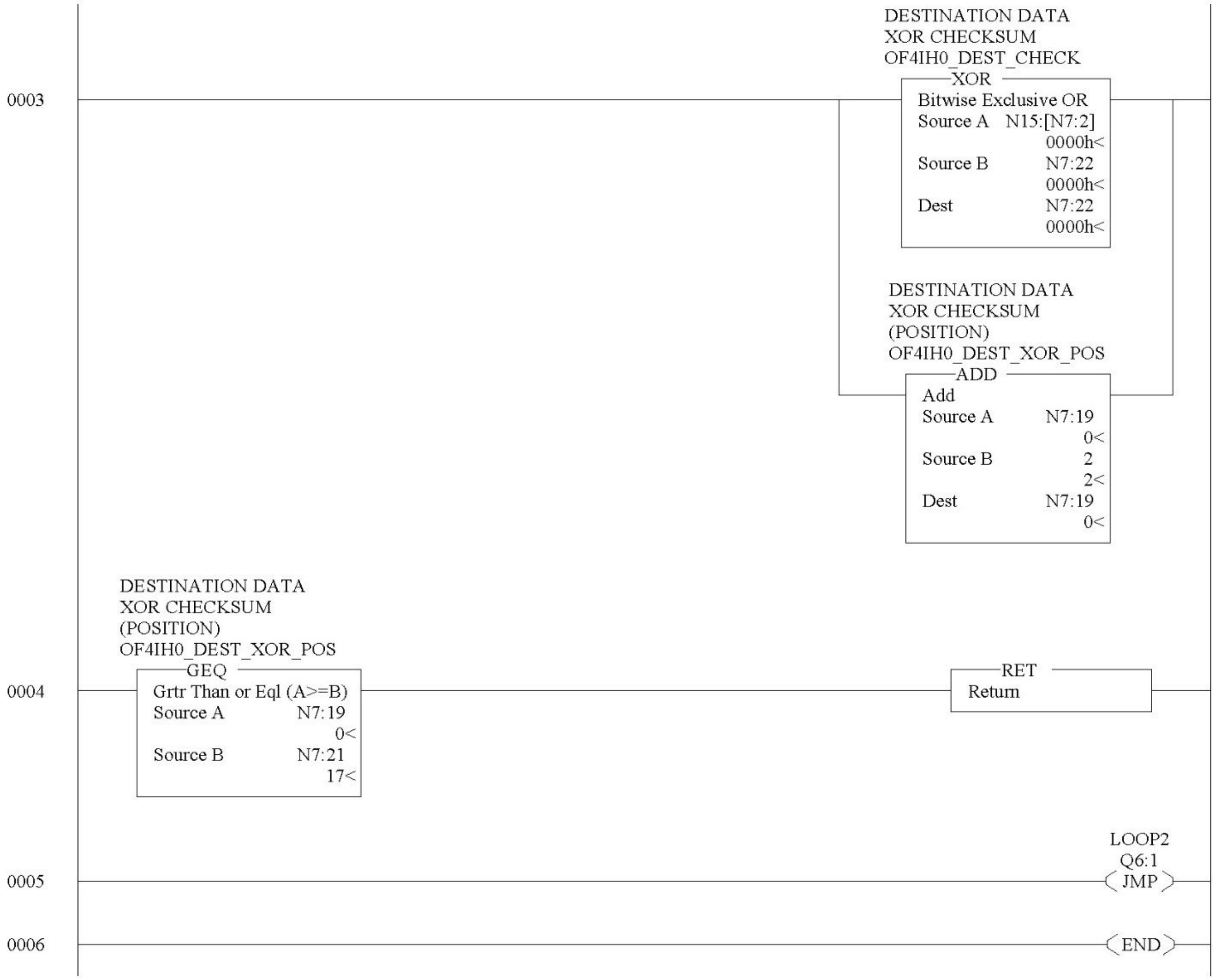


Figure 7-9b.


0003
DESTINATION DATA
XOR CHECKSUM
OF4IH0_DEST_CHECK
XOR
Bitwise Exclusive OR
Source A N15:[N7:2]
0000h<
Source B N7:22
0000h<
Dest N7:22
0000h<
DESTINATION DATA
XOR CHECKSUM
(POSITION)
OF4IH0_DEST_XOR_POS
ADD
Add
Source A N7:19
0<
Source B 2
2<
Dest N7:19
0<
0004
DESTINATION DATA
XOR CHECKSUM
(POSITION)
OF4IH0_DEST_XOR_POS
GEQ
Grtr Than or Eql (A>=B)
Source A N7:19
0<
Source B N7:21
17<
RET
Return
0005
LOOP2
Q6:1
JMP
0006
END

7.2.6 HART_MSG Routine

This routine composes HART messages that will be sent to the module/field transmitter. This routine is called from the MAIN routine.

Figure 7-10a. HART Message

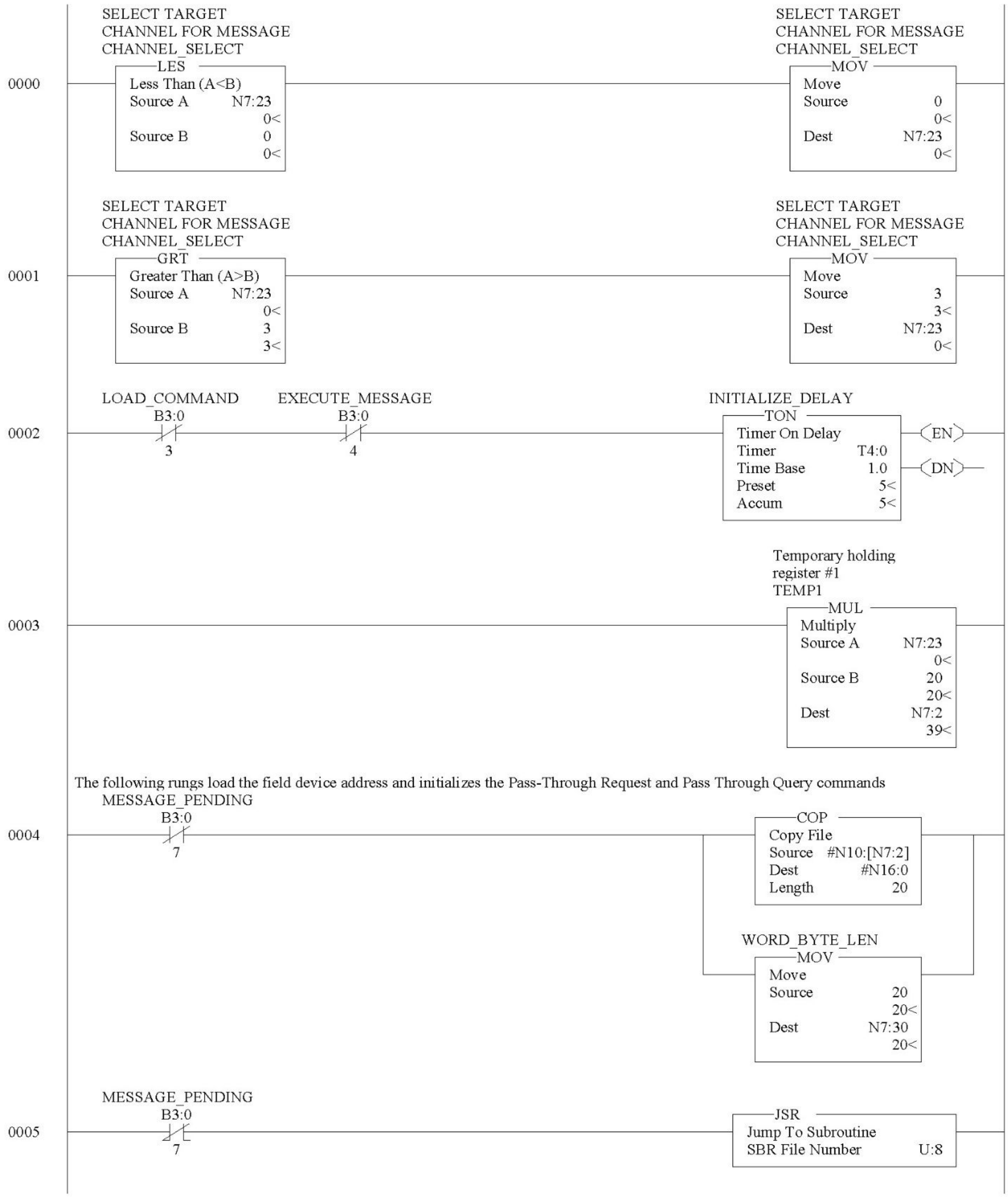


Figure 7-10b.


0006
MESSAGE_PENDING
B3:0
7
COP
Copy File
Source #N15:0
Dest #N7:40
Length 40

Figure 7-10c.

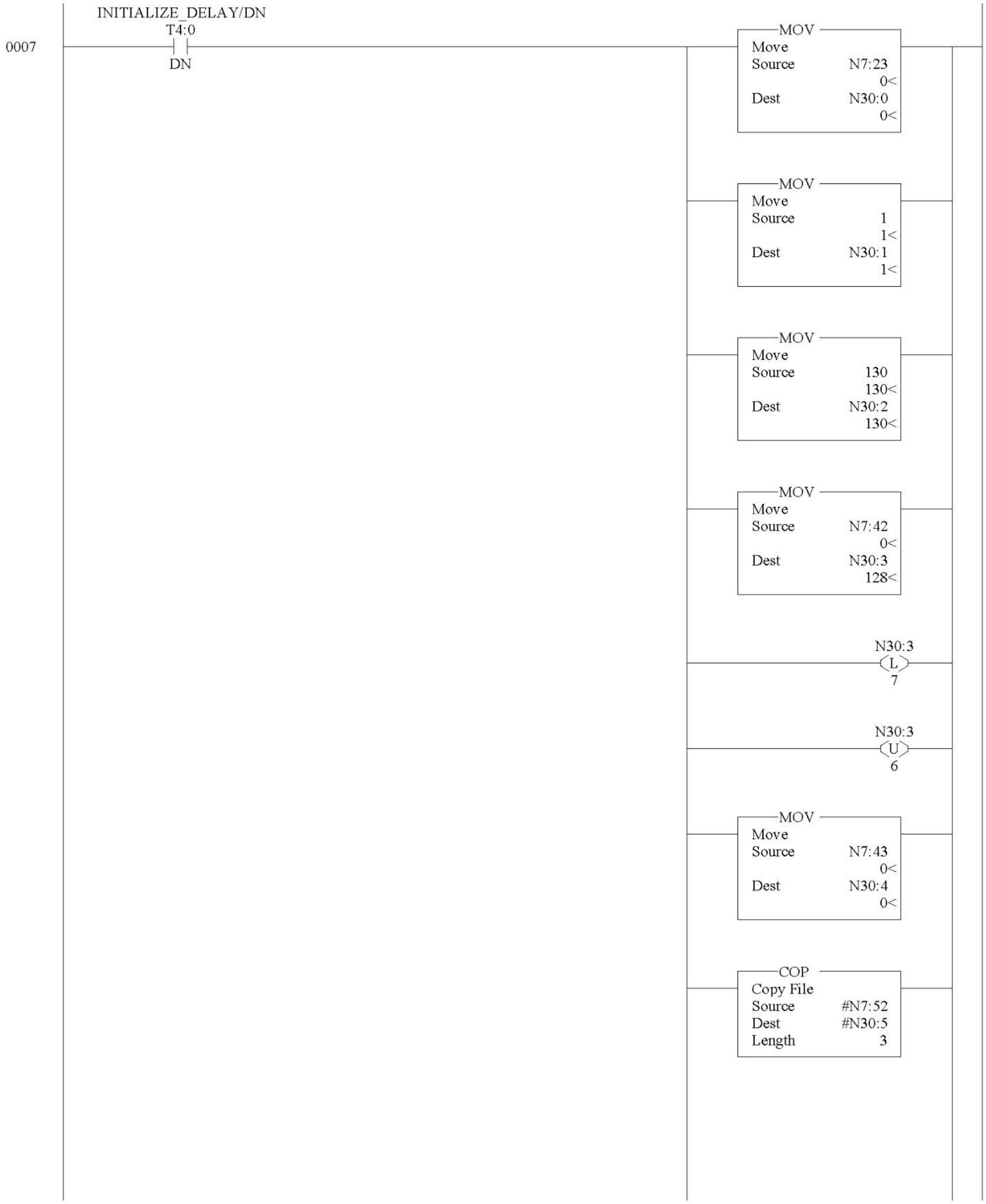

INITIALIZE_DELAY/DN
T4:0
0007
DN
MOV
Move
Source N7:23
0<
Dest N30:0
0<
MOV
Move
Source 1
1<
Dest N30:1
1<
MOV
Move
Source 130
130<
Dest N30:2
130<
MOV
Move
Source N7:42
0<
Dest N30:3
128<
N30:3
L
7
N30:3
U
6
MOV
Move
Source N7:43
0<
Dest N30:4
0<
COP
Copy File
Source #N7:52
Dest #N30:5
Length 3

Figure 7-10d.

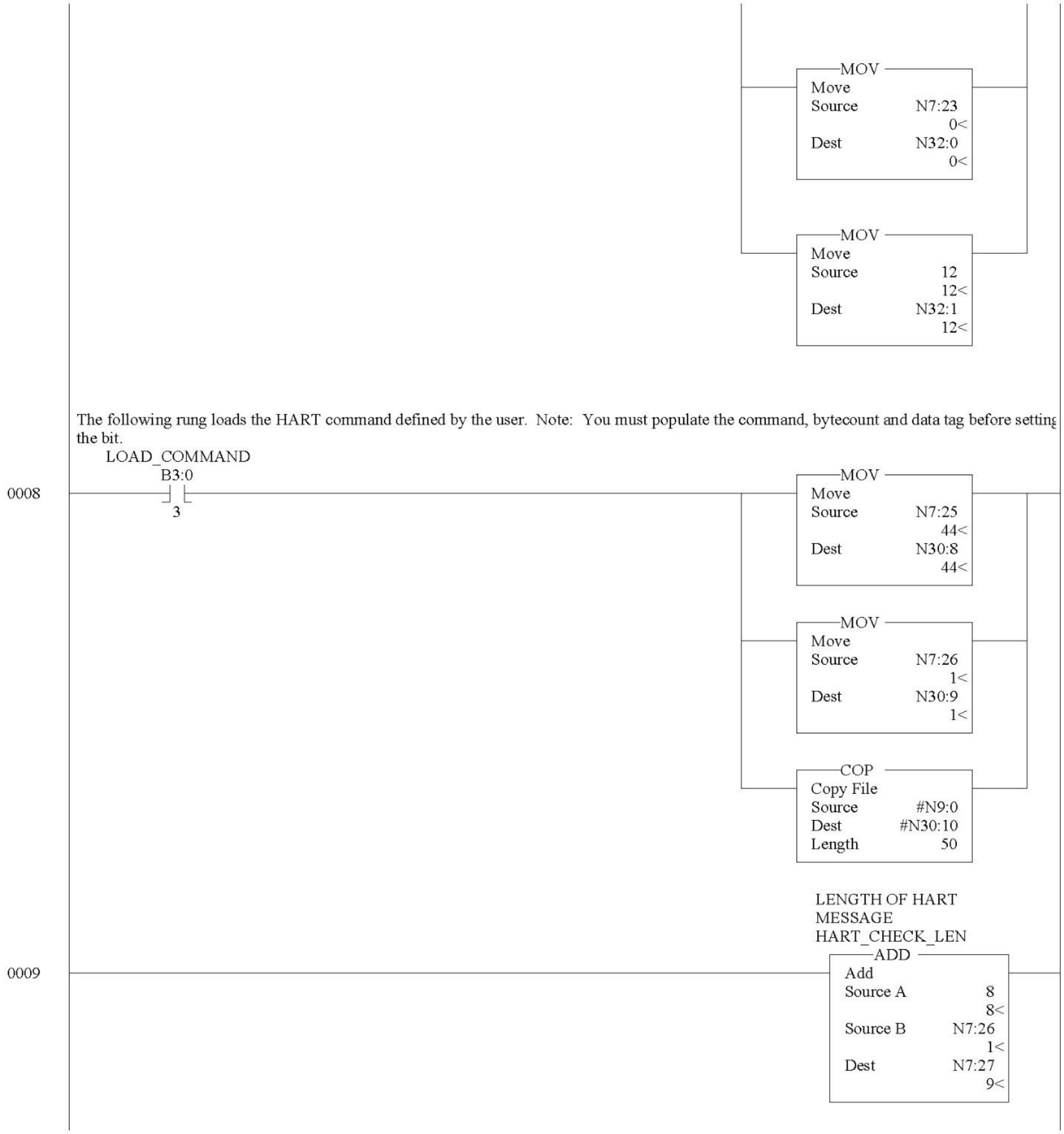

MOV
Move
Source N7:23
0<
Dest N32:0
0<
MOV
Move
Source 12
12<
Dest N32:1
12<
The following rung loads the HART command defined by the user. Note: You must populate the command, bytecount and data tag before setting the bit.
LOAD_COMMAND
B3:0
0008
3
MOV
Move
Source N7:25
44<
Dest N30:8
44<
MOV
Move
Source N7:26
1<
Dest N30:9
1<
COP
Copy File
Source #N9:0
Dest #N30:10
Length 50
LENGTH OF HART
MESSAGE
HART_CHECK_LEN
0009
ADD
Add
Source A 8
8<
Source B N7:26
1<
Dest N7:27
9<

Figure 7-10e.

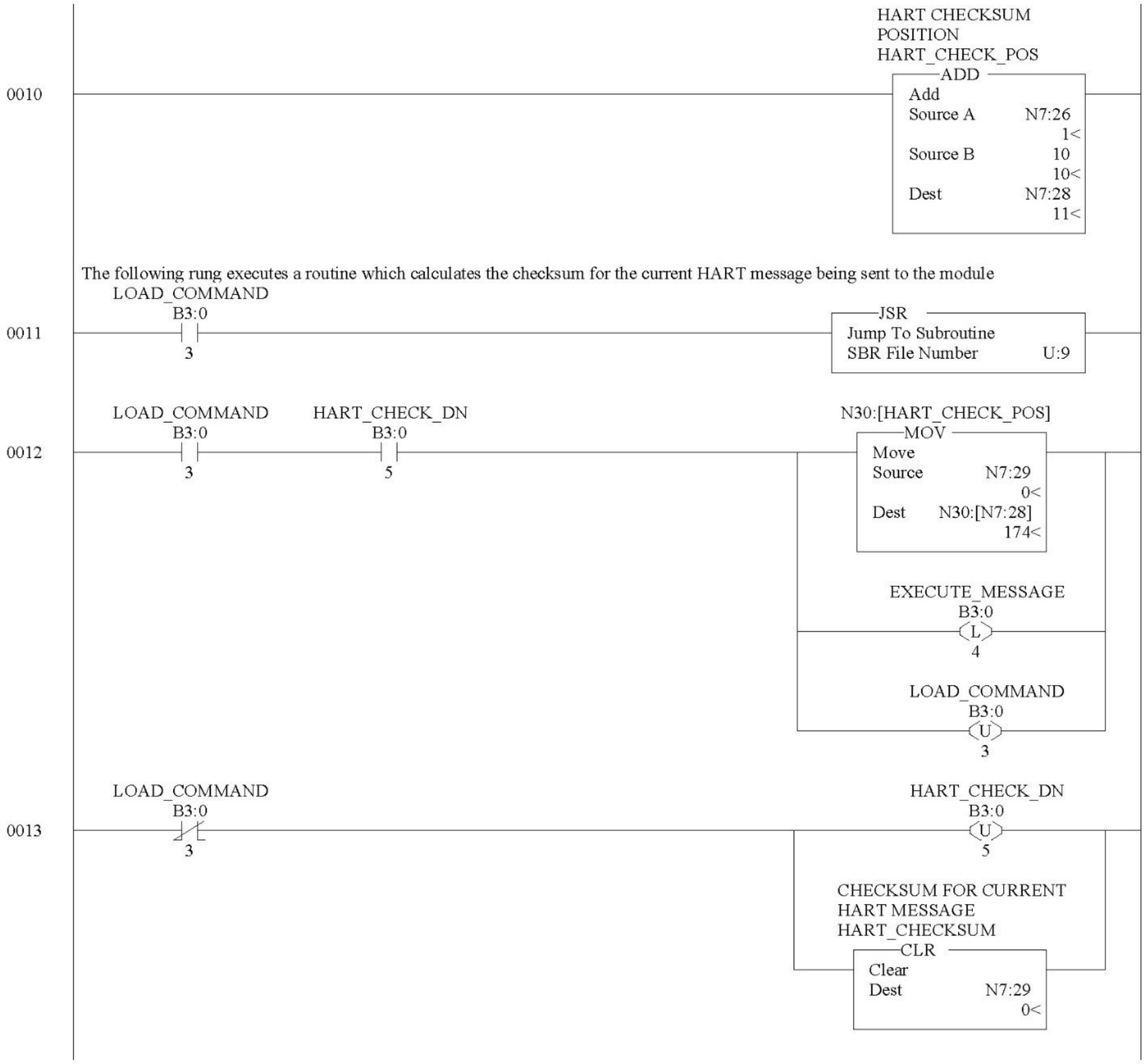

HART CHECKSUM
POSITION
HART_CHECK_POS
0010
ADD
Add
Source A N7:26
1<
Source B 10
10<
Dest N7:28
11<
The following rung executes a routine which calculates the checksum for the current HART message being sent to the module
LOAD_COMMAND
B3:0
0011
3
JSR
Jump To Subroutine
SBR File Number U:9
LOAD_COMMAND
B3:0
HART_CHECK_DN
B3:0
N30:[HART_CHECK_POS]
0012
3
5
MOV
Move
Source N7:29
0<
Dest N30:[N7:28]
174<
EXECUTE_MESSAGE
B3:0
L
4
LOAD_COMMAND
B3:0
U
3
LOAD_COMMAND
B3:0
0013
3
HART_CHECK_DN
B3:0
U
5
CHECKSUM FOR CURRENT
HART MESSAGE
HART_CHECKSUM
CLR
Clear
Dest N7:29
0<

Figure 7-10f.

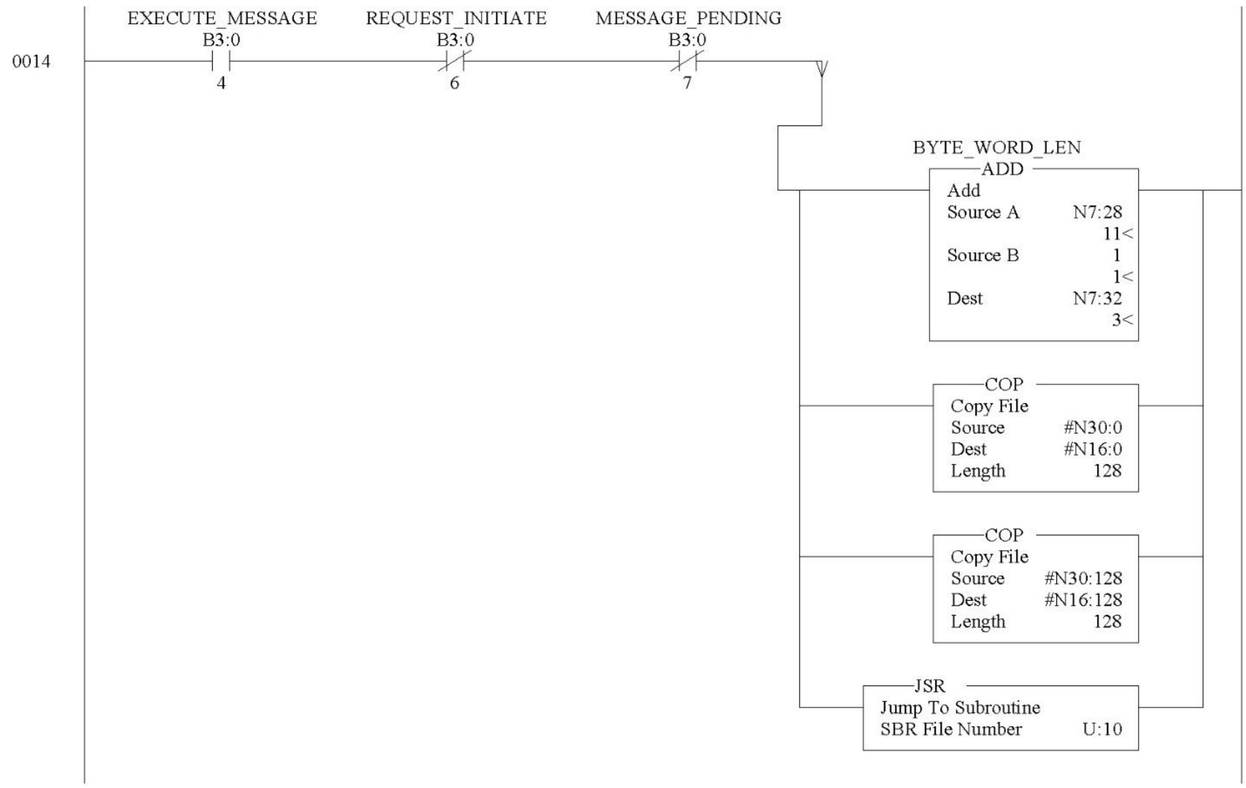

0014
EXECUTE_MESSAGE
B3:0
4
REQUEST_INITIATE
B3:0
6
MESSAGE_PENDING
B3:0
7
BYTE_WORD_LEN
ADD
Add
Source A N7:28
11<
Source B 1
1<
Dest N7:32
3<
COP
Copy File
Source #N30:0
Dest #N16:0
Length 128
COP
Copy File
Source #N30:128
Dest #N16:128
Length 128
JSR
Jump To Subroutine
SBR File Number U:10

Figure 7-10g.

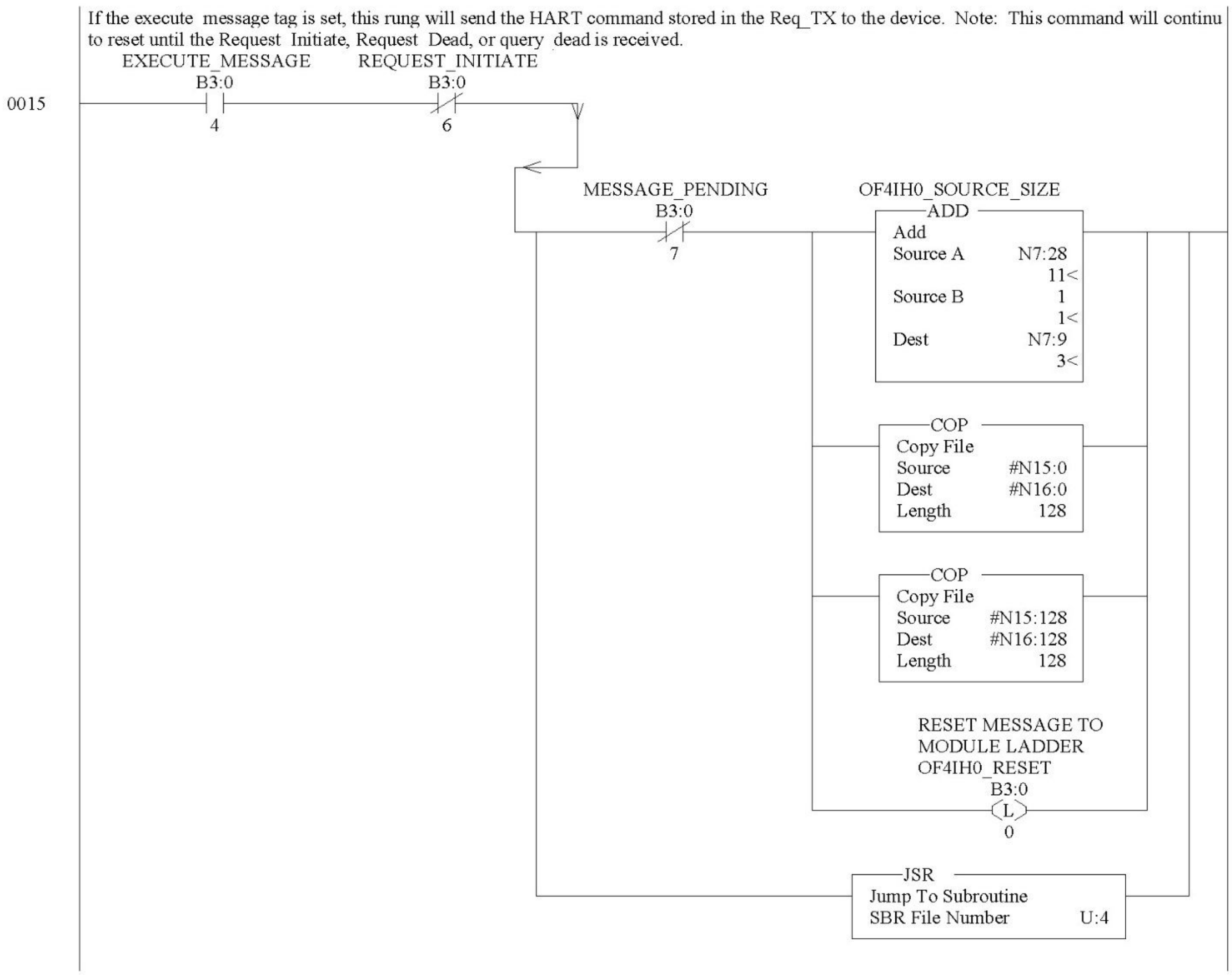

If the execute message tag is set, this rung will send the HART command stored in the Req_TX to the device. Note: This command will continu
to reset until the Request Initiate, Request Dead, or query dead is received.
EXECUTE_MESSAGE
B3:0
4
REQUEST_INITIATE
B3:0
6
0015
MESSAGE_PENDING
B3:0
7
OF4IH0_SOURCE_SIZE
ADD
Add
Source A N7:28
11<
Source B 1
1<
Dest N7:9
3<
COP
Copy File
Source #N15:0
Dest #N16:0
Length 128
COP
Copy File
Source #N15:128
Dest #N16:128
Length 128
RESET MESSAGE TO
MODULE LADDER
OF4IH0_RESET
B3:0
L
0
JSR
Jump To Subroutine
SBR File Number U:4

Figure 7-10h.

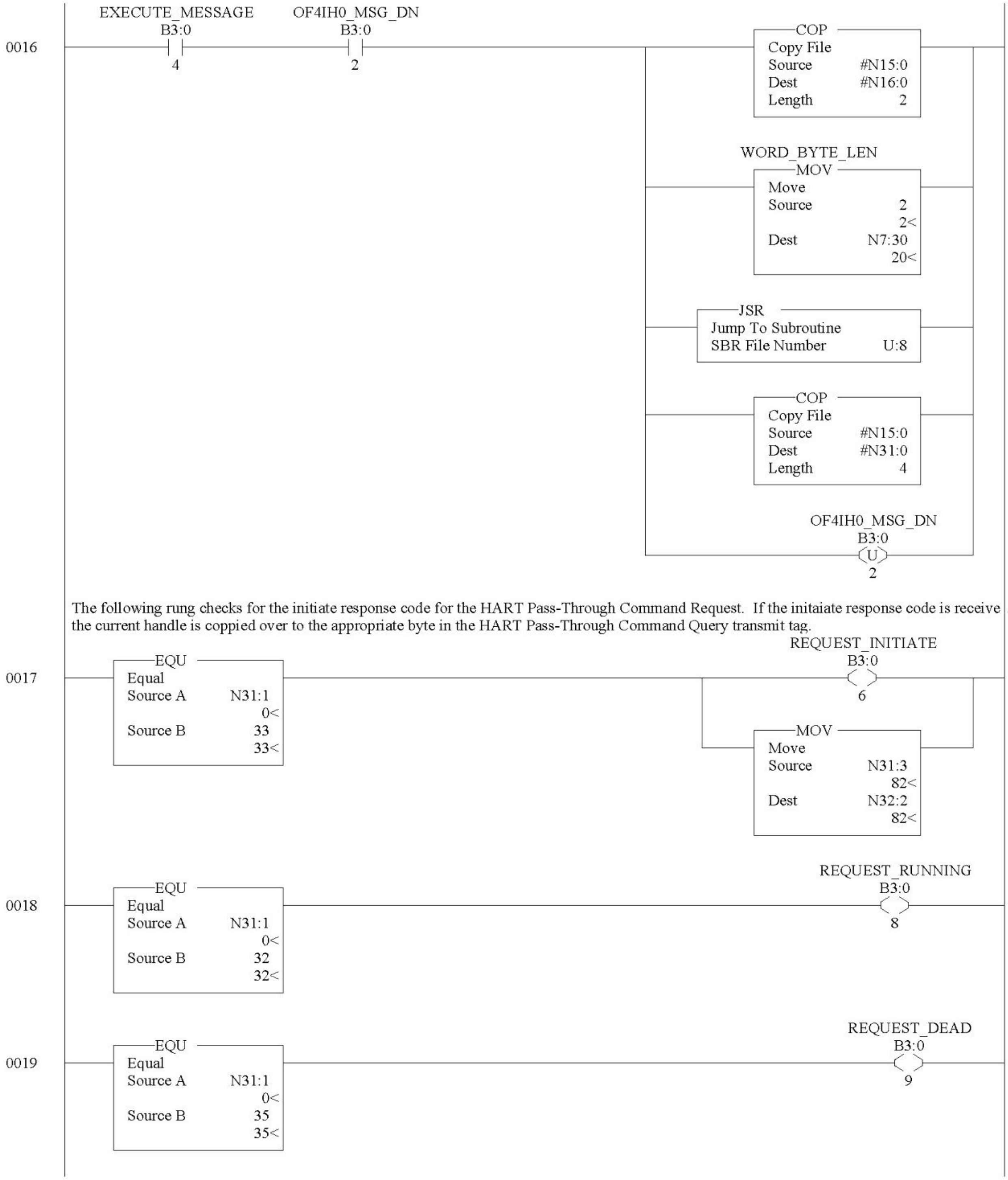
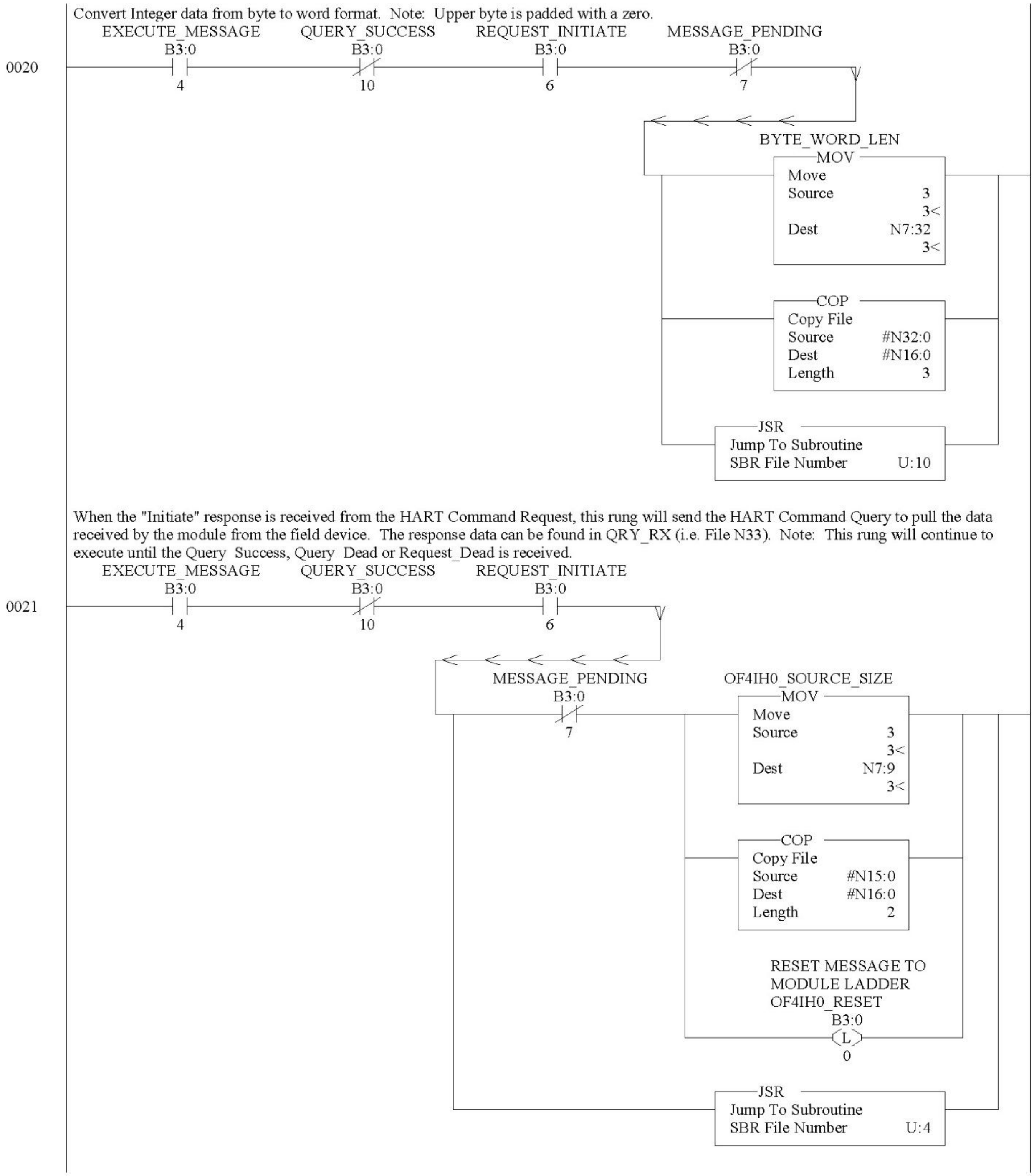

0016
EXECUTE_MESSAGE
B3:0
4
OF4IH0_MSG_DN
B3:0
2
COP
Copy File
Source #N15:0
Dest #N16:0
Length 2
WORD_BYTE_LEN
MOV
Move
Source 2
2<
Dest N7:30
20<
JSR
Jump To Subroutine
SBR File Number U:8
COP
Copy File
Source #N15:0
Dest #N31:0
Length 4
OF4IH0_MSG_DN
B3:0
U
2
The following rung checks for the initiate response code for the HART Pass-Through Command Request. If the initaiate response code is receive
the current handle is coppied over to the appropriate byte in the HART Pass-Through Command Query transmit tag.
0017
EQU
Equal
Source A N31:1
0<
Source B 33
33<
REQUEST_INITIATE
B3:0
6
MOV
Move
Source N31:3
82<
Dest N32:2
82<
0018
EQU
Equal
Source A N31:1
0<
Source B 32
32<
REQUEST_RUNNING
B3:0
8
0019
EQU
Equal
Source A N31:1
0<
Source B 35
35<
REQUEST_DEAD
B3:0
9

Figure 7-10i.



Convert Integer data from byte to word format. Note: Upper byte is padded with a zero.
0020
EXECUTE_MESSAGE
B3:0
4
QUERY_SUCCESS
B3:0
10
REQUEST_INITIATE
B3:0
6
MESSAGE_PENDING
B3:0
7
BYTE_WORD_LEN
MOV
Move
Source 3
3<
Dest N7:32
3<
COP
Copy File
Source #N32:0
Dest #N16:0
Length 3
JSR
Jump To Subroutine
SBR File Number U:10
When the "Initiate" response is received from the HART Command Request, this rung will send the HART Command Query to pull the data received by the module from the field device. The response data can be found in QRY_RX (i.e. File N33). Note: This rung will continue to execute until the Query Success, Query Dead or Request_Dead is received.
0021
EXECUTE_MESSAGE
B3:0
4
QUERY_SUCCESS
B3:0
10
REQUEST_INITIATE
B3:0
6
MESSAGE_PENDING
B3:0
7
OF4IH0_SOURCE_SIZE
MOV
Move
Source 3
3<
Dest N7:9
3<
COP
Copy File
Source #N15:0
Dest #N16:0
Length 2
RESET MESSAGE TO
MODULE LADDER
OF4IH0_RESET
B3:0
L
0
JSR
Jump To Subroutine
SBR File Number U:4

Figure 7-10j.

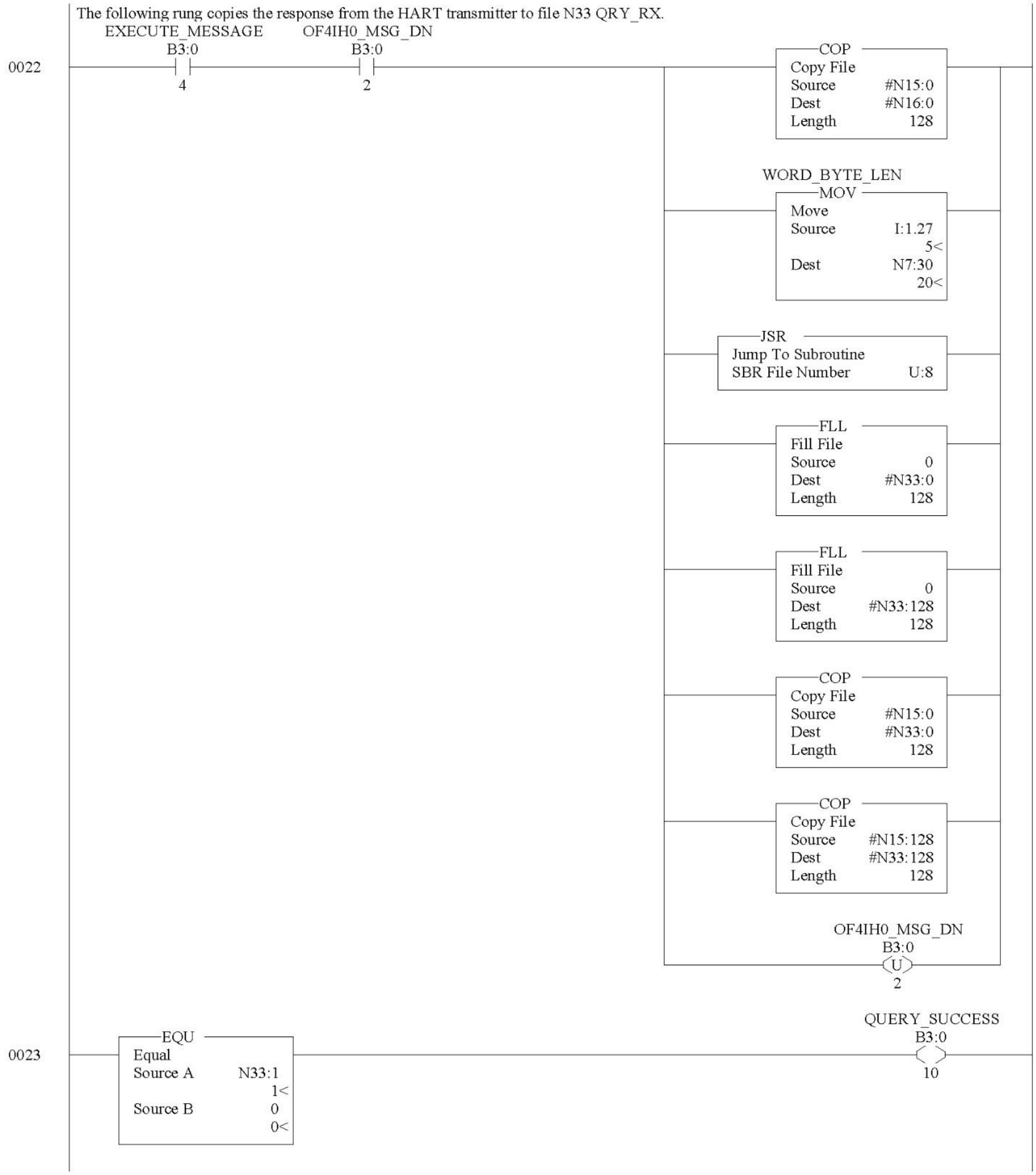
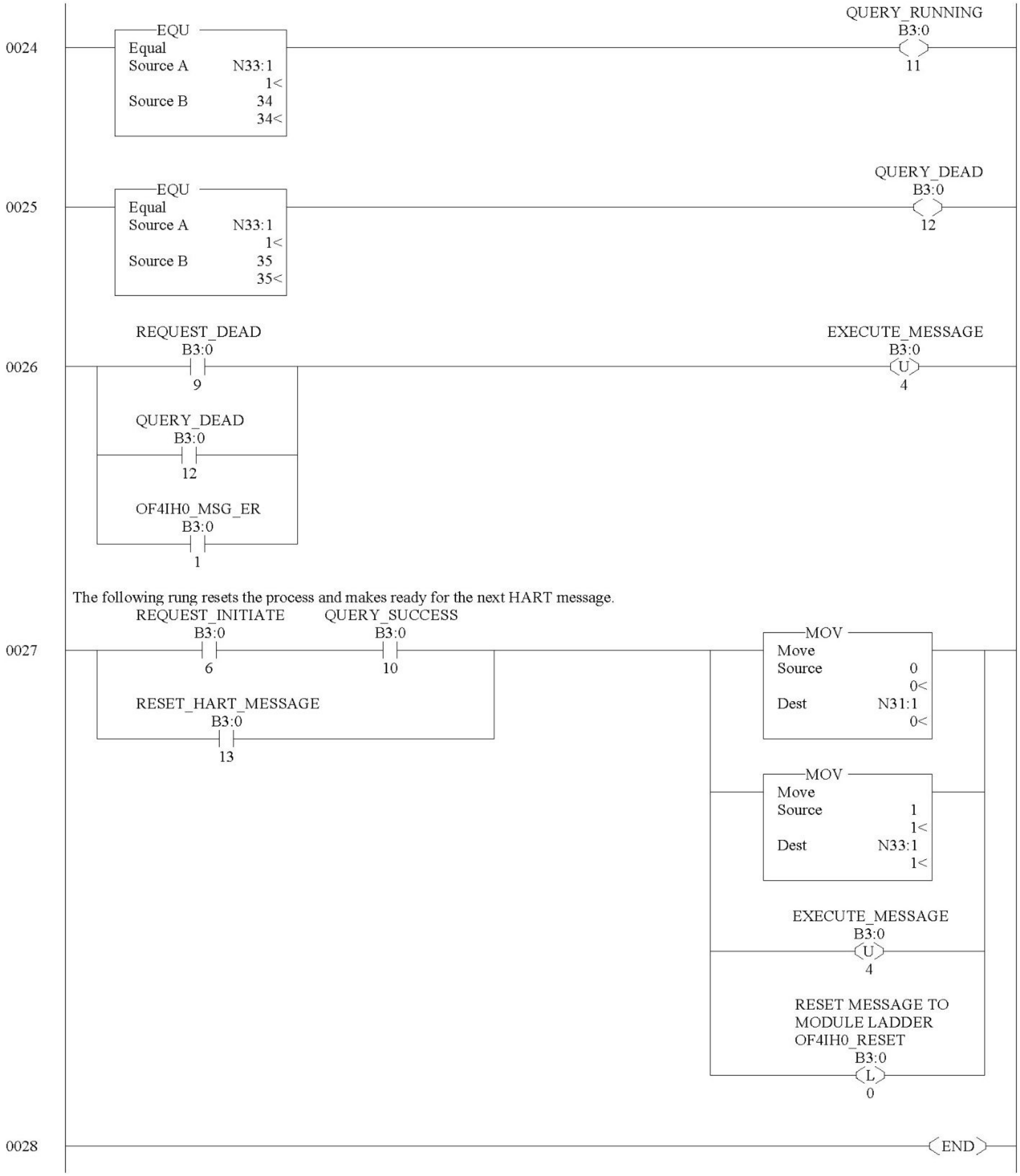

The following rung copies the response from the HART transmitter to file N33 QRY_RX.
0022
EXECUTE_MESSAGE
B3:0
4
OF4IH0_MSG_DN
B3:0
2
COP
Copy File
Source #N15:0
Dest #N16:0
Length 128
WORD_BYTE_LEN
MOV
Move
Source I:1.27
5<
Dest N7:30
20<
JSR
Jump To Subroutine
SBR File Number U:8
FLL
Fill File
Source 0
Dest #N33:0
Length 128
FLL
Fill File
Source 0
Dest #N33:128
Length 128
COP
Copy File
Source #N15:0
Dest #N33:0
Length 128
COP
Copy File
Source #N15:128
Dest #N33:128
Length 128
OF4IH0_MSG_DN
B3:0
U
2
0023
EQU
Equal
Source A N33:1
1<
Source B 0
0<
QUERY_SUCCESS
B3:0
10

Figure 7-10k.


QUERY_RUNNING
B3:0
11
0024
EQU
Equal
Source A N33:1
1<
Source B 34
34<
QUERY_DEAD
B3:0
12
0025
EQU
Equal
Source A N33:1
1<
Source B 35
35<
REQUEST_DEAD
B3:0
9
EXECUTE_MESSAGE
B3:0
U
4
0026
QUERY_DEAD
B3:0
12
OF4IH0_MSG_ER
B3:0
1
The following rung resets the process and makes ready for the next HART message.
REQUEST_INITIATE
B3:0
6
QUERY_SUCCESS
B3:0
10
0027
RESET_HART_MESSAGE
B3:0
13
MOV
Move
Source 0
0<
Dest N31:1
0<
MOV
Move
Source 1
1<
Dest N33:1
1<
EXECUTE_MESSAGE
B3:0
U
4
RESET MESSAGE TO
MODULE LADDER
OF4IH0_RESET
B3:0
L
0
0028
END

7.2.7 WORD_BYTE Routine

Converts word data to its byte equivalent. This routine is called from the HART_MSG routine.

Figure 7-11a. Word to Byte

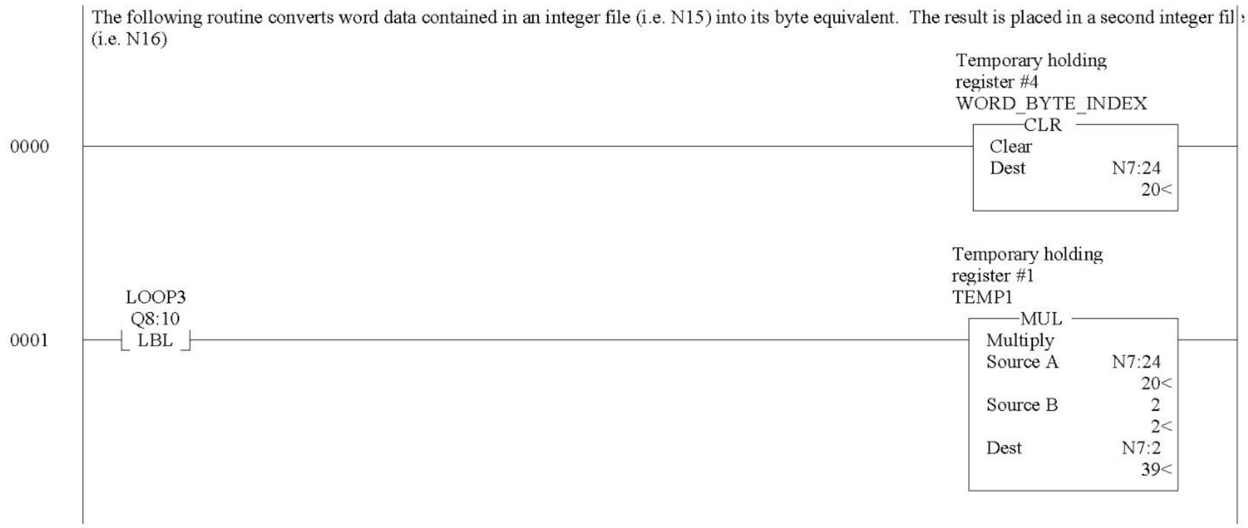


Figure 7-11b.

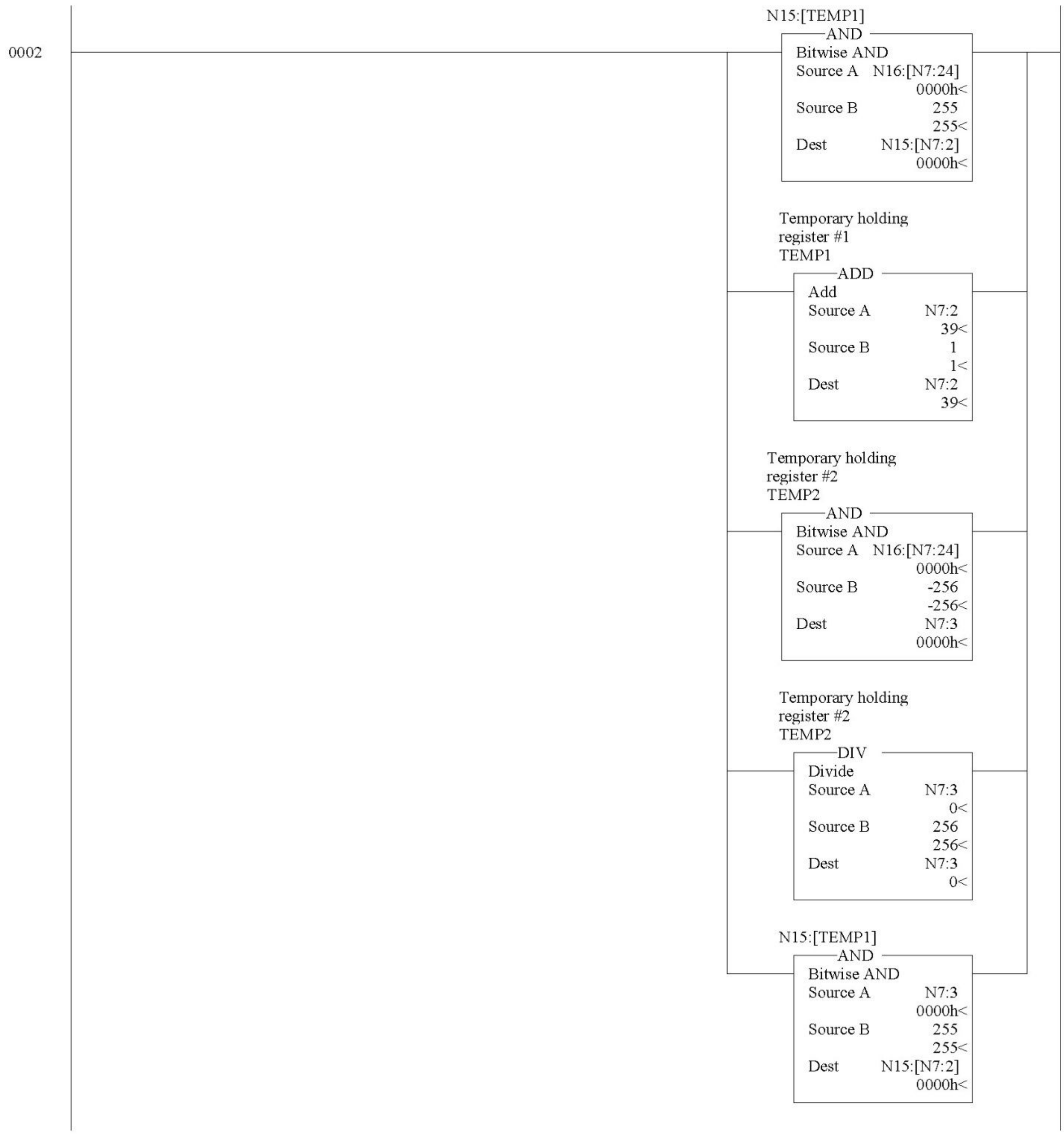
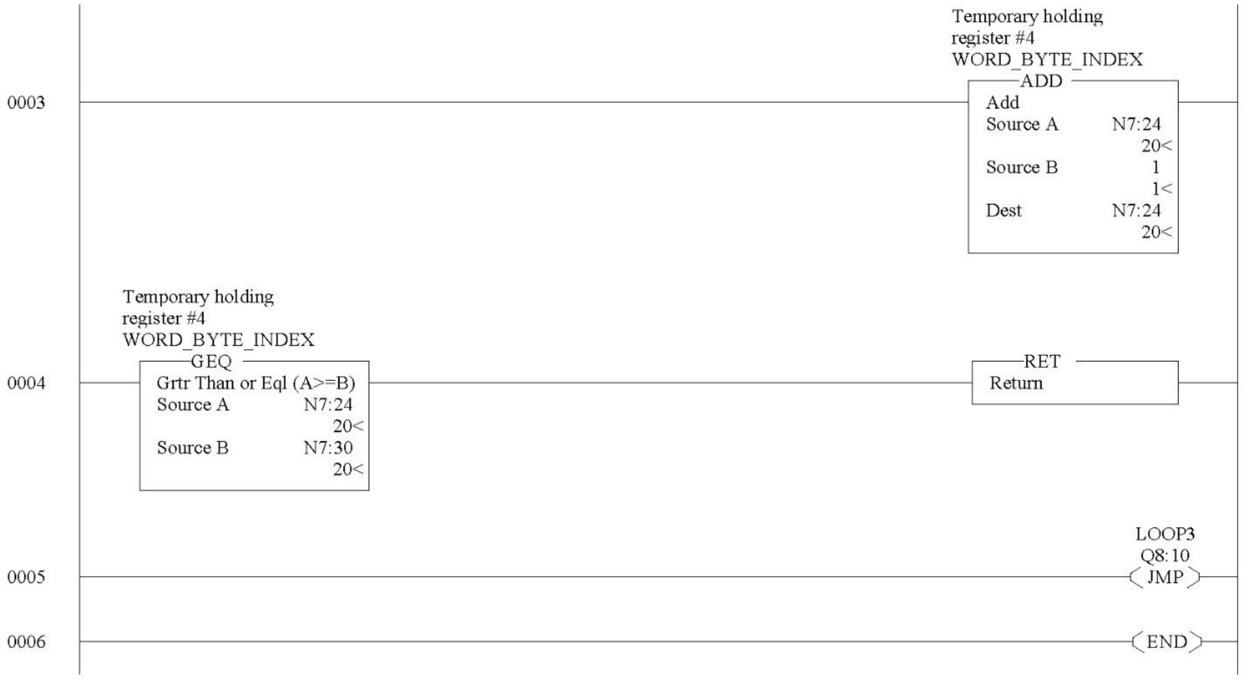

0002
N15:[TEMP1]
AND
Bitwise AND
Source A N16:[N7:24]
0000h<
Source B 255
255<
Dest N15:[N7:2]
0000h<
Temporary holding
register #1
TEMP1
ADD
Add
Source A N7:2
39<
Source B 1
1<
Dest N7:2
39<
Temporary holding
register #2
TEMP2
AND
Bitwise AND
Source A N16:[N7:24]
0000h<
Source B -256
-256<
Dest N7:3
0000h<
Temporary holding
register #2
TEMP2
DIV
Divide
Source A N7:3
0<
Source B 256
256<
Dest N7:3
0<
N15:[TEMP1]
AND
Bitwise AND
Source A N7:3
0000h<
Source B 255
255<
Dest N15:[N7:2]
0000h<

Figure 7-11c.


Temporary holding
register #4
WORD_BYTE_INDEX
ADD
Add
Source A N7:24
20<
Source B 1
1<
Dest N7:24
20<
0003
Temporary holding
register #4
WORD_BYTE_INDEX
GEQ
Grtr Than or Eql (A>=B)
Source A N7:24
20<
Source B N7:30
20<
0004
RET
Return
LOOP3
Q8:10
JMP
0005
END
0006

7.2.8 HART_CHECK Routine

Calculates the checksum for the HART message being sent to the module/field device. This routine is called from the HART_MSG routine.

Figure 7-12a. HART Checksum

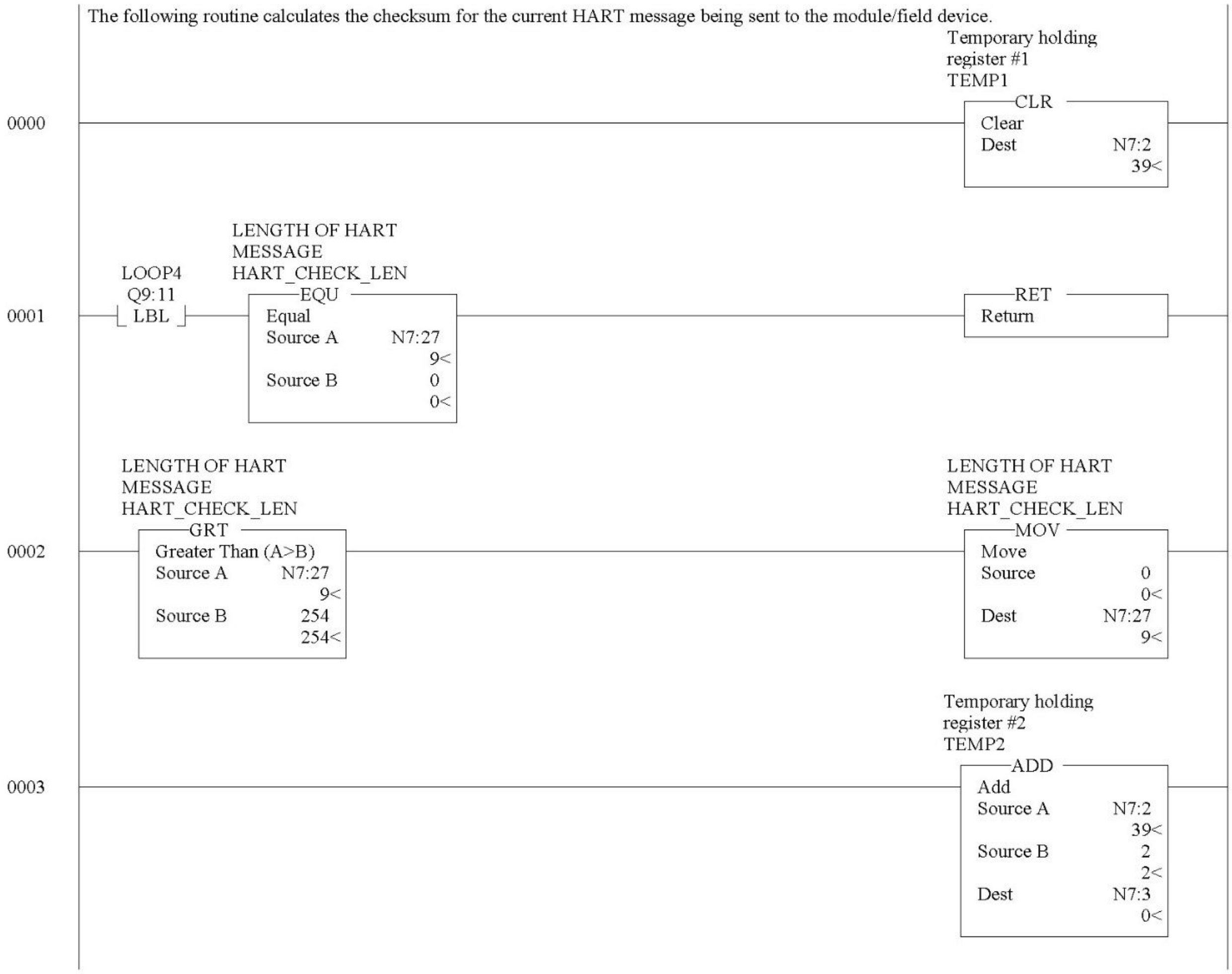
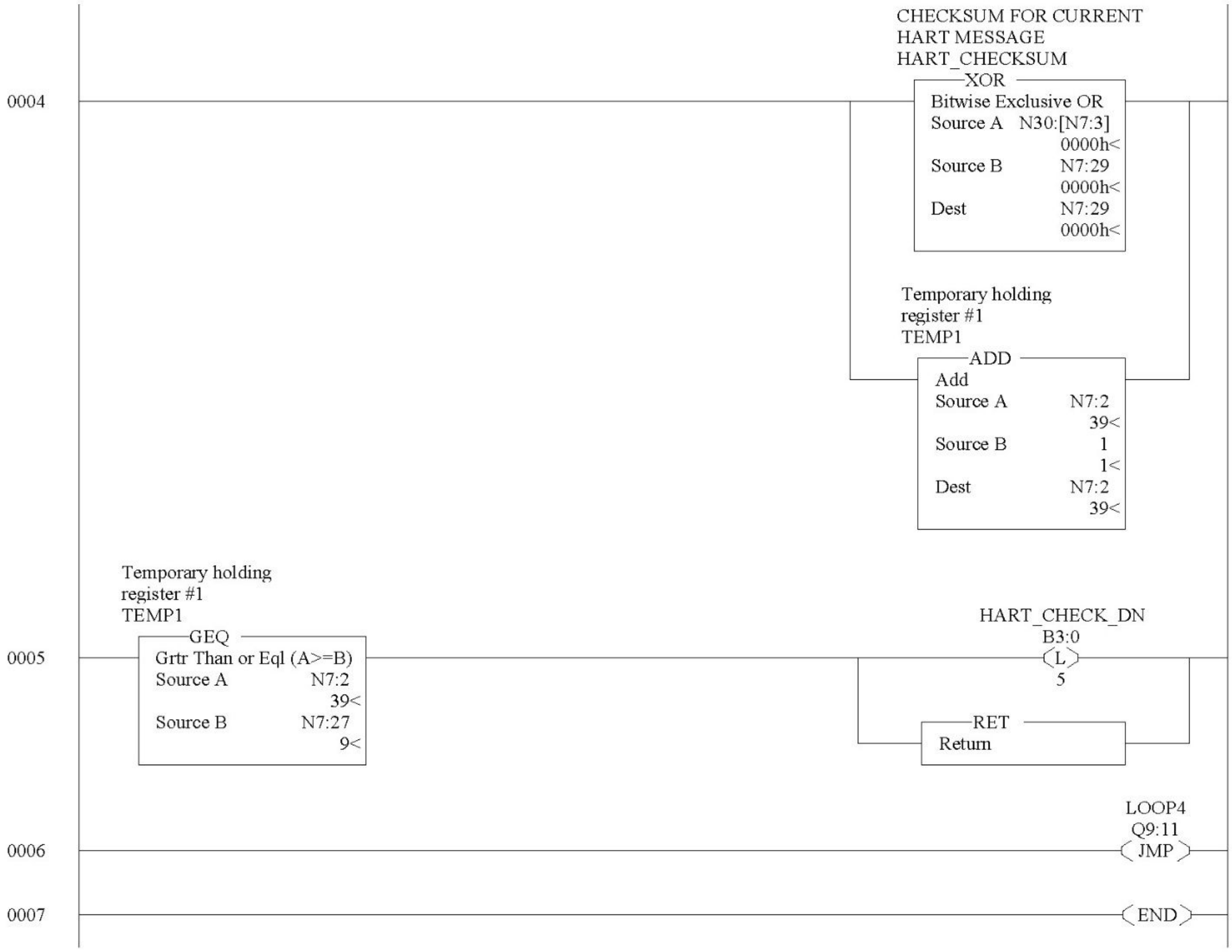


Figure 7-12b.


CHECKSUM FOR CURRENT
HART MESSAGE
HART_CHECKSUM
XOR
Bitwise Exclusive OR
Source A N30:[N7:3]
0000h<
Source B N7:29
0000h<
Dest N7:29
0000h<
0004
Temporary holding
register #1
TEMP1
ADD
Add
Source A N7:2
39<
Source B 1
1<
Dest N7:2
39<
Temporary holding
register #1
TEMP1
GEQ
Grtr Than or Eql (A>=B)
Source A N7:2
39<
Source B N7:27
9<
0005
HART_CHECK_DN
B3:0
L
5
RET
Return
0006
LOOP4
Q9:11
JMP
0007
END

7.2.9 BYTE_WORD Routine

Converts byte data to its word equivalent. This routine is called by the HART_MSG routine.

Figure 7-13a. Byte to Word

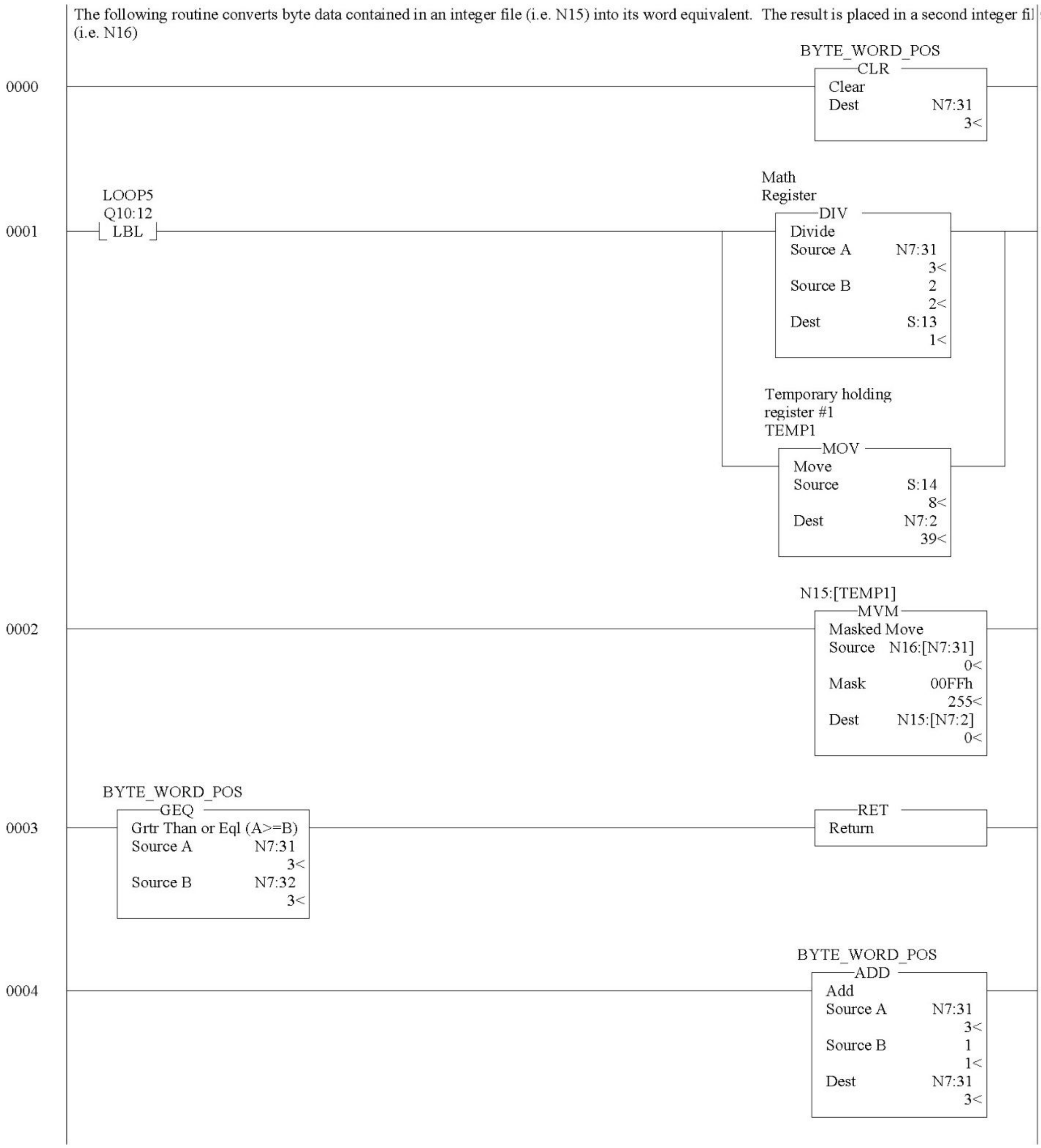
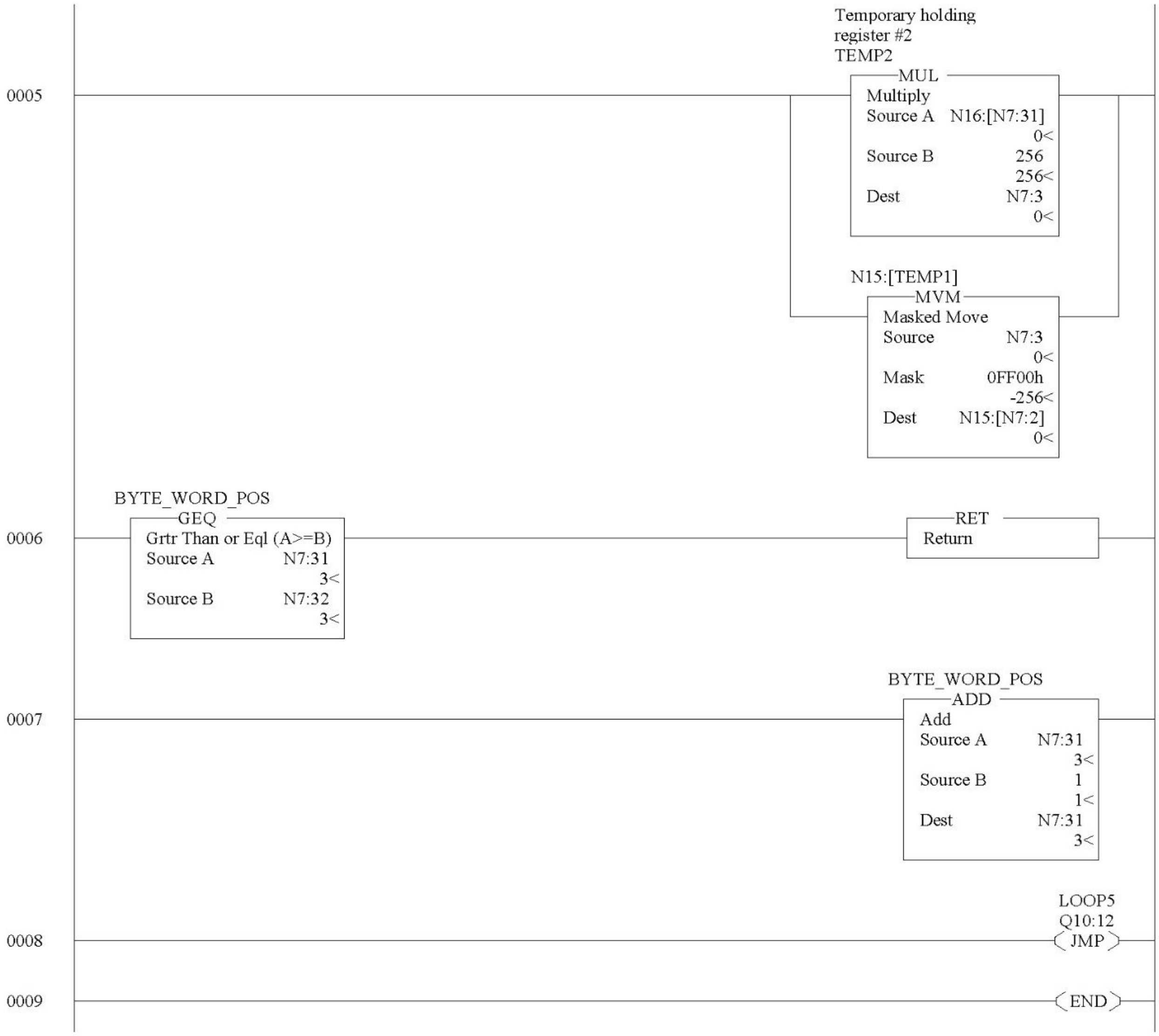


Figure 7-13b.


Temporary holding
register #2
TEMP2
MUL
Multiply
Source A N16:[N7:31]
0<
Source B 256
256<
Dest N7:3
0<
0005
N15:[TEMP1]
MVM
Masked Move
Source N7:3
0<
Mask 0FF00h
-256<
Dest N15:[N7:2]
0<
BYTE_WORD_POS
GEQ
Grtr Than or Eql (A>=B)
Source A N7:31
3<
Source B N7:32
3<
0006
RET
Return
BYTE_WORD_POS
ADD
Add
Source A N7:31
3<
Source B 1
1<
Dest N7:31
3<
0007
LOOP5
Q10:12
JMP
0008
END
0009

Chapter 8 Diagnostics and Troubleshooting

This chapter describes troubleshooting the isolated HART output module. This chapter contains information on:

- Safety considerations while troubleshooting
- Internal diagnostics during module operation
- Module error codes
- Contacting Spectrum Controls, Inc. for technical assistance

Section 8.1 Safety Considerations

Safety considerations are an important element of proper troubleshooting procedures. Actively thinking about the safety of yourself and others, as well as the condition of your equipment, is of primary importance.

The following sections describe several safety concerns you should be aware of when troubleshooting your control system.

NOTE	Never reach into a machine to actuate a switch because unexpected motion can occur and cause injury. Remove all electrical power at the main power disconnect switches before checking electrical connections or inputs/outputs causing machine motion.

8.1.1 Indicator Lights

When the green LED on the module is illuminated, it indicates that power is applied to the module and that it has passed its internal tests.

8.1.2 Stand Clear of Equipment

When troubleshooting any system problem, have all personnel remain clear of the equipment. The problem could be intermittent, and sudden unexpected machine motion could occur. Have someone ready to operate an emergency stop switch in case it becomes necessary to shut off power.

8.1.3 Program Alteration

There are several possible causes of alteration to the user program, including extreme environmental conditions, Electromagnetic Interference (EMI), improper grounding, improper wiring connections, and unauthorized tampering. If you suspect a program has been altered, check it against a previously saved master program.

8.1.4 Safety Circuits

Circuits installed on the machine for safety reasons, like over-travel limit switches, stop push buttons, and interlocks, should always be hard-wired 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.

Section 8.2 Module Operation vs. Channel Operation

The module performs diagnostic operations at both the module level and the channel level. Module-level operations include functions such as power-up, configuration, and communication with a 1769 bus master, such as a MicroLogix 1500 controller, 1769-ADN DeviceNet Adapter, or CompactLogix controller.

Channel-level operations describe channel related functions, such as data conversion and over- or under-range detection.

Internal diagnostics are performed at both levels of operation. When detected, module error conditions are immediately indicated by the module status LED. Both module hardware and channel configuration error conditions are reported to the controller. Channel over-range or under-range conditions are visible in the module's input data table. Module hardware errors are typically reported in the controller's I/O status file. Refer to your controller manual for details.

Section 8.3 Power-up Diagnostics

At module power-up, a series of internal diagnostic tests are performed. If these diagnostic tests are not successfully completed, the module status LED remains off and a module error is reported to the controller.

Table 8-1. LED Status

If module status LED is:	Indicated condition	Corrective action:
On	Proper Operation	No action required
Off	Module Fault	Cycle power. If condition persists, replace the module. Call your local distributor or Spectrum Controls for assistance.

Section 8.4 Channel Diagnostics

When an output channel is enabled, the module performs a diagnostic check to see that the channel has been properly configured. In addition, the channel is tested on every scan for configuration errors, over-range and under-range conditions.

8.4.1 Invalid Channel Configuration Detection

Whenever a channel configuration word is improperly defined, the module reports an error. See Table 8-4 for a description of module errors.

8.4.2 Over or Under-Range Detection

Whenever a channel is commanded to an output level outside of the defined operating range, an over-range or under-range error is indicated in input data word 1. Possible causes of an out-of-range condition include:

- The output device is faulty.
- The field wiring may be faulty.
- The *low clamp* or *high clamp* values have been exceeded.

Section 8.5 Non-critical vs. Critical Module Errors

Non-critical module errors are typically recoverable. Channel errors (over-range or under-range errors) are non-critical. Non-critical error conditions are indicated in the module input data table.

Critical module errors are conditions that may prevent normal or recoverable operation of the system. When these types of errors occur, the system typically leaves the run or program mode of operation until the error can be dealt with. Critical module errors are indicated in Table 8-4. Extended Fault Codes.

Section 8.6 Module Error Definition Table

Analog module errors are expressed in two fields as four-digit Hex format with the most significant digit as “don’t care” and irrelevant. The two fields are “Module Error” and “Extended Error Information”. The structure of the module error data is shown below.

Table 8-2. Module Error Table

"Don't Care" Bits			Module Error		Extended Error Information										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hex Digit 4				Hex Digit 3				Hex Digit 2				Hex Digit 1			

8.6.1 Module Error Field

The purpose of the module error field is to classify module errors into three distinct groups, as described in the table below. The type of error determines what kind of information exists in the extended error information field. These types of module errors are typically reported in the controller's I/O status file. Refer to your controller manual for details.

Table 8-3. Module Error Types

Error Type	Module Error Field Value Bits 11 through 9 (binary)	Description
No errors	000	No error is present. The extended error field holds no additional information.
Hardware Errors	001	General and specific hardware error codes are specified in the extended error information field.
Configuration Errors	010	Module-specific error codes are indicated in the extended error field. These error codes correspond to options that you can change directly. For example, the input range or input filter selection.

8.6.2 Extended Error Information Field

Check the extended error information field when a non-zero value is present in the module error field. Depending upon the value in the module error field, the extended error information field can contain error codes that are module-specific or common to all 1769 analog modules.

NOTE	If no errors are present in the module error field, the extended error information field is set to zero.

Hardware Errors

General or module-specific hardware errors are indicated by module error code 001. See Table 8-4. Extended Fault Codes.

Configuration Errors

If you set the fields in the configuration file to invalid or unsupported values, the module generates a critical error.

Table 8-4. Extended Fault Codes lists the possible module-specific configuration error codes defined for the module.

Section 8.7
Error Codes

The table below explains the extended error code.

Table 8-4. Extended Fault Codes

Error Type	Hex Equivalent	Module Error Code	Extended Error Information Code	Error Description
No error	X000	000	0 0000 0000	No error
Hardware-Specific Error	X216	001	0 0001 0110	Watchdog reset error
	X220	001	0 0010 0000	Critical code failure
	X221	001	0 0010 0001	Failed calibration/critical EEPROM failure
Module-Specific Configuration Error	X401	010	0 0000 0001	Channel 0 Invalid Output Range
	X402	010	0 0000 0010	Channel 1 Invalid Output Range
	X403	010	0 0000 0011	Channel 2 Invalid Output Range
	X404	010	0 0000 0100	Channel 3 Invalid Output Range
	X405	010	0 0000 0101	Channel 0 Invalid Data Format
	X406	010	0 0000 0110	Channel 1 Invalid Data Format
	X407	010	0 0000 0111	Channel 2 Invalid Data Format
	X408	010	0 0000 1000	Channel 3 Invalid Data Format
	X409	010	0 0000 1001	Channel 0 Invalid Fault Value
	X40A	010	0 0000 1010	Channel 1 Invalid Fault Value
	X40B	010	0 0000 1011	Channel 2 Invalid Fault Value
	X40C	010	0 0000 1100	Channel 3 Invalid Fault Value
	X40D	010	0 0000 1101	Channel 0 Invalid Idle Value
	X40E	010	0 0000 1110	Channel 1 Invalid Idle Value
	X40F	010	0 0000 1111	Channel 2 Invalid Idle Value
	X410	010	0 0001 0000	Channel 3 Invalid Idle Value
	X411	010	0 0001 0001	Channel 0 Invalid Clamps
	X412	010	0 0001 0010	Channel 1 Invalid Clamps
	X413	010	0 0001 0011	Channel 2 Invalid Clamps
	X414	010	0 0001 0100	Channel 3 Invalid Clamps
	X415	010	0 0001 0101	Channel 0 Invalid Ramp Rate
	X416	010	0 0001 0110	Channel 1 Invalid Ramp Rate
	X417	010	0 0001 0111	Channel 2 Invalid Ramp Rate
	X418	010	0 0001 1000	Channel 3 Invalid Ramp Rate
	X419	010	0 0001 1001	Channel 0 Config word 0 Illegal bits set

Error Type	Hex Equivalent	Module Error Code	Extended Error Information Code	Error Description
	X41A	010	0 0001 1010	Channel 1 Config word 0 Illegal bits set
	X41B	010	0 0001 1011	Channel 2 Config word 0 Illegal bits set
	X41C	010	0 0001 1100	Channel 3 Config word 0 Illegal bits set
	X41D	010	0 0001 1101	Channel 0 Config word 1 Illegal bits set
	X41E	010	0 0001 1110	Channel 1 Config word 1 Illegal bits set
	X41F	010	0 0001 1111	Channel 2 Config word 1 Illegal bits set
	X420	010	0 0010 0000	Channel 3 Config word 1 Illegal bits set
	X421	010	0 0010 0001	Channel 0 Illegal HART Passthrough rate.
	X422	010	0 0010 0010	Channel 1 Illegal HART Passthrough rate.
	X423	010	0 0010 0011	Channel 2 Illegal HART Passthrough rate.
	X424	010	0 0010 0100	Channel 3 Illegal HART Passthrough rate.

Section 8.8 Module Inhibit Function

Some controllers support the module inhibit function. See your controller manual for details.

Whenever the 1769sc-OF4IH module is inhibited, the module continues to provide information about changes at its outputs to the 1769 CompactBus master (for example, a CompactLogix controller).

Section 8.9 Getting Technical Assistance

Note that your module contains electronic components which are susceptible to damage from electrostatic discharge (ESD). An electrostatic charge can accumulate on the surface of ordinary plastic wrapping or cushioning material. **In the unlikely event that the module should need to be returned to Spectrum Controls, please ensure that the unit is enclosed in approved ESD packaging (such as static-shielding / metallized bag or black conductive container).** Spectrum Controls reserves the right to void the warranty on any unit that is improperly packaged for shipment.

For further information or assistance, please contact your local distributor, or call the Spectrum Controls Technical Support at:

USA - 425-746-9481

Section 8.10 Declaration of Conformity

Available upon request

Appendix A
Module Specifications

Electrical Specifications

<table>
<tr><td>Specification</td><td colspan="2">Description</td></tr>
<tr><td>Number of Channels</td><td colspan="2">4 single isolated outputs with HART</td></tr>
<tr><td>Output Ranges</td><td colspan="2">0 to 20.0 mA and 4 to 20 mA</td></tr>
<tr><td>Output Load</td><td colspan="2">750 ohms maximum</td></tr>
<tr><td>Resolution</td><td colspan="2">16 bits over full design range (0-21.24 mA)</td></tr>
<tr><td>Full Scale Settling Time
without HART enabled</td><td colspan="2"><45 ms to 95% of full scale</td></tr>
<tr><td>Full Scale Settling Time
with HART enabled</td><td colspan="2"><70 ms to 95% of full scale</td></tr>
<tr><td>Analog Output Update Rate</td><td colspan="2">10 ms update of all 8 channels</td></tr>
<tr><td>Differential Nonlinearity</td><td colspan="2">±0.05% full scale based on 4-20 mA range.</td></tr>
<tr><td>Output Ripple</td><td colspan="2">±0.05%, or better full scale 0-50 kHz</td></tr>
<tr><td>Repeatability</td><td colspan="2">±0.05% full scale</td></tr>
<tr><td>Max Inductive Load</td><td colspan="2">0.1 mH</td></tr>
<tr><td>Output Impedance</td><td colspan="2">> 1 Mohm</td></tr>
<tr><td rowspan="3">Accuracy</td><td>Temperature</td><td>Accuracy
(% Full Scale)</td></tr>
<tr><td>25 °C</td><td>0.35%</td></tr>
<tr><td>0 °C to 60 °C</td><td>0.55%</td></tr>
<tr><td>Calibration</td><td colspan="2">Modules will be factory-calibrated.</td></tr>
<tr><td>Electrical Isolation (one minute)</td><td colspan="2">710 VDC field wiring to backplane
710 VDC channel-to-channel isolation
500 VDC field wiring to chassis ground</td></tr>
<tr><td>Resistive Load</td><td colspan="2">50- to 750-ohm drive, current</td></tr>
<tr><td>Maximum Backplane Current draw</td><td colspan="2">180 mA at 5 VDC, 200 mA at 24 VDC</td></tr>
<tr><td>Thermal Dissipation</td><td colspan="2">4.2 W at 21.0 mA current all channels with 250-ohm minimum load.</td></tr>
<tr><td>Maximum Slot Distance</td><td colspan="2">8 for backplane +24 V</td></tr>
<tr><td>Overvoltage Protection</td><td colspan="2">+24V continuous on any current output</td></tr>
<tr><td>Power on/off channel glitch</td><td colspan="2">Under ±1 V for 2 ms. Target: ±0.5 V spike for <5 ms</td></tr>
<tr><td>Open and Short Circuit Protection</td><td colspan="2">Yes, continuous. (EN61131-2 requirement)</td></tr>
</table>

Environmental Specifications

Standard	Description	Specification
IEC 610068-2-27	Shock Operating	30 g, 11 ms, three shock in each direction on each axis
test Ea		
IEC 610068-2-27	Shock Non-Operating	50 g, 11 ms, three shock in each direction on each axis
test Ea		
IEC 610068-2-6	Vibration Operating	10 Hz-57 Hz 0.3 mm
test Fc		57 Hz-500 Hz 3 g
IEC 610068-2-31 or 32	Free-Fall	1 m height, onto concrete
Procedure 1		
IEC 610068-2-2	Dry Heat withstanding	<50% RH, - 85 °C, 96-hr cycle
test Bb	Non-Operating	
IEC 610068-2-2	Dry Heat withstanding	<50% RH, 60 °C, 6 × 8 hr cycle
test Be	Operating	
IEC 610068-2-1	Cold Withstanding	<50% RH, -40 °C, 96-hr cycle
test Ab	Non-Operating	
IEC 610068-2-1	Cold Withstanding	<50% RH, 0 °C, 6 × 8 hr cycle
test Ae	Operating	
IEC 610068-2-30	Damp Heat, Cyclic	95% RH, 25 °C - 60 °C, 12 hr × 12 hr cycle
test Db		6 cycles

Regulatory Compliance

Certifications (when product is marked)[21] **cULus**	**UL** Listed for Class I, Division 2 Group A, B, C, D Hazardous Locations, certified for U.S. and Canada. See UL File E180101. **UL** Listed Industrial Control Equipment, certified for U.S. and Canada. See UL File E140954. **Ex** European Union 2014/34/EU
	EN 60079-7:2015+A1:2018; Potentially Explosive Atmospheres, Protection “ec” (Zone 2) II 3 G Ex ec IIC T4 Gc
	EN 60079-0: ATEX General Requirements
	Certificate UL 20 ATEX 2403X
CCC	GB/T3836.1-2021, GB/T3836.3-2021
	GBEx 2021312310000325
	GBEx 2021312310000343
	CE European Union 2014/30/EU EMC Directive, compliant with: EN 61000-6-4; Industrial Emissions EN 61000-6-2; Industrial Immunity EN 61131-2; Programmable Controllers (Clause 8, Zone A & B)
UKCA	Electromagnetic Compatibility Regulations 2016
	BS EN 61131-2, BS EN 61000-6-4, BS EN 61000-6-2
	Equipment and Protective Systems Intended for use in Potentially Explosive Atmospheres Regulations 2016
	BS EN 60079-0, BS EN 60079-7
CMIM	Arrêté ministériel n° 6404-15 du 29 ramadan 1436 (16 juillet 2015)
	NM EN 61131-2, NM EN 61000-6-4, NM EN 61000-6-2

[21] For the latest up-to-date information, see the Product Certification link at www.spectrumcontrols.com for Declarations of Conformity, Certificates and other certification details.

Appendix B
HART Universal and Common Practice Commands

COMMAND NUMBER AND FUNCTION	DATA IN COMMAND (TYPE)			DATA IN REPLY		(TYPE)
0 Read unique identifier		none		Byte 0	"254" (expansion)	
				Byte 1	manufacturer identification code	
				Byte 2	mfr's device type code	
				Byte 3	number of preambles	
				Byte 4	universal command revision	
				Byte 5	transmitter-specific cmd revision	
				Byte 6	software revision	
				Byte 7	hardware revision	(H)
				Byte 8	device function flags	(B)
				Byte 9-11	device ID number	
1 Read primary variable		none		Byte 0	PV units code	
				Byte 1-4	primary variable	(F)
2 Read current and percent of range		none		Byte 0-3	current (mA)	(F)
				Byte 4-7	percent of range	(F)
3 Read current and four (predefined) dynamic variables		none		Byte 0-3	current (mA)	(F)
				Byte 4	PV units code	
				Byte 5-8	primary variable	(F)
				Byte 9	SV units code	
				Byte 10-13	second variable	(F)
				Byte 14	TV units code	
				Byte 15-18	third variable	(F)
				Byte 19	FV units code	
				Byte 20-23	fourth variable	(F)
6 Write polling addr.	Byte 0	polling address			as in command	
11 Read unique ident. associated with tag	Byte 0-5	tag	(A)		as command 0	
12 Read message		none		Byte 0-23	message	(A)
13 Read tag, descriptor, date		none		Byte 0-5	tag	(A)
				Byte 6-17	descriptor	(A)
				Byte 18-20	date	(D)
14 Read PV sensor information		none		Byte 0-2	sensor serial number	
				Byte 3	units code for sensor limits and min span	
				Byte 4-7	upper sensor limit	(F)
				Byte 8-11	lower sensor limit	(F)
				Byte 12-15	minimum span	(F)
15 Read output information		none		Byte 0	alarm select code	
				Byte 1	transfer function code	
				Byte 2	PV/range units code	
				Byte 3-6	upper range value	(F)
				Byte 7-10	lower range v	(F)
				Byte 11-14	damping value (sec)	(F)
				Byte 15	write-protect code	
				Byte 16	private-label distributor code	
16 Read final assembly number		none		Byte 0-2	final assembly number	
17 Write message	Byte 0-23	message	(A)			
18 Write tag, descriptor, date	Byte 0-5	tag	(A)		as in command	
	Byte 6-17	descriptor	(A)			
	Byte 18-20	date	(D)			
19 Write final assembly number	Byte 0-2	final assembly number			as in command	

COMMAND NUMBER AND FUNCTION	DATA IN COMMAND	(TYPE)	DATA IN REPLY	(TYPE)
33 Read transmitter variables	Byte 0 transm. var. code for slot 0 Byte 1 transm. var. code for slot 1 Byte 2 transm. var. code for slot 2 Byte 3 transm. var. code for slot 3 Byte 7 units code for slot 1 (truncated after last requested code) Byte 12 transmitter var. code for slot 2 Byte 13 units code for slot 2 Byte 14-17 variable for slot 2 Byte 18 transmitter var. code for slot 3 Byte 19 units code for slot 3 Byte 20-23 variable for slot 3 (truncated after last requested variable)		Byte 0 transmitter var. code for slot 0 Byte 1 units code for slot 0 Byte 2-5 variable for slot 0 Byte 6 transmitter var. code for slot 1 Byte 8-11 variable for slot 1	(F) (F) (F) (F)
34 Write damping value	Byte 0-3 damping value (sec)	(F)	as in command	
35 Write range values	Byte 0 range units code Byte 1-4 upper range value Byte 5-8 lower range value	(F) (F)	as in command	
36 Set upper range value (= push SPAN button)	none		none	
37 Set lower range value (= push ZERO button)	none		none	
38 Reset "configuration changed" flag	none		none	
39 EEPROM control	Byte 0 EEPROM control code (0 = burn EEPROM, 1 = copy EEPROM to RAM)		as in command	
40 Enter/exit fixed current mode	Byte 0-3 current (mA) (0 = exit the mode)	(F)	as in command	
41 Perform transmitter self test	none		none	
42 Perform master reset	none		none	
43 Set PV zero	none		none	
44 Write PV units	Byte 0 PV units code		as in command	
45 Trim DAC zero	Byte 0-3 measured current (mA)	(F)	as in command	
46 Trim DAC gain	Byte 0-3 measured current (mA)	(F)	as in command	
47 Write transfer function	Byte 0 transfer function code		as in command	
48 Read additional transmitter status	none		Byte 0-24 additional status	
49 Write PV sensor serial number	Byte 0-2 sensor serial number		as in command	
50 Read dynamic variable assignments	none		Byte 0 PV transmitter variable code Byte 1 SV transmitter variable code Byte 2 TV transmitter variable code Byte 3 FV transmitter variable code	

COMMAND NUMBER AND FUNCTION	DATA IN COMMAND (TYPE)	DATA IN REPLY (TYPE)
51 Write dynamic variable assignments	Byte 0 PV transmitter variable code Byte 1 SV transmitter variable code Byte 2 TV transmitter variable code Byte 3 FV transmitter variable code	as in command
52 Set transmitter variable zero	Byte 0 transmitter variable code	as in command
53 Write transmitter variable units	Byte 0 transmitter variable code Byte 1 transmitter var. units code	as in command
54 Read transmitter variable information	Byte 0 transmitter variable code	Byte 0 transmitter variable code Byte 1-3 transm. var. sensor serial number Byte 4 transm. var. limits units code Byte 5-8 transm. variable upper limit (F) Byte 9-12 transm. variable lower limit (F) Byte 13-16 transm. var. damping value (sec)(F)
55 Write transmitter variable damping value	Byte 0 transmitter variable code Byte 1-4 transmitter variable damping value (sec (F)	as in command
56 Write transmitter variable sensor serial number	Byte 0 transmitter variable code Byte 1-3 transmitter variable sensor serial number	as in command
57 Read unit tag, descriptor, date	none	Byte 0-5 unit tag (A) Byte 6-17 unit descriptor (A) Byte 18-20 unit date (D)
58 Write unit tag, descriptor, date	Byte 0-5 unit tag (A) Byte 6-17 unit descriptor (A) Byte 18-20 unit date (D)	as in command
59 Write number of response preambles	Byte 0 number of response preambles	as in command
108 Write burst mode command number	Byte 0 burst mode command number	as in command
109 Burst mode control	Byte 0 burst mode control code (0 = exit, 1 = enter)	as in command
110 Read all dynamic variables	none	Byte 0 PV units code Byte 1-4 PV value (F) Byte 5 SV units code Byte 6-9 SV value (F) Byte 10 TV units code Byte 11-14 TV value (F) Byte 15 FV units code Byte 16-19 FV value (F)

Data types:
A ASCII string (packed 4 characters per 3 bytes)
B Bit-mapped flags (bit 0 = multisensor device; bit 1 = EEPROM control required)
D Date (day, month, year—1900)
F Floating point (4 bytes IEEE 754)
H Integers xxxxx yyy (xxxxx = hardware rev., yyy = physical signalling code)
Unmarked items are 8-, 16- or 24-bit integers

First byte:

BIT 7 = 1COMMUNICATION ERRORS:	
Bit 6	parity error
Bit 5	overrun error
Bit 4	framing error
Bit 3	checksum error
Bit 2	(reserved)
Bit 1	rx buffer overflow
Bit 0	(undefined)

Second byte:

Bit 7	
Bit 6	
Bit 5	
Bit 4	all 0
Bit 3	
Bit 2	
Bit 1	
Bit 0	

First byte:

BIT 7 = 0COMMAND ERRORS:	
Bits 6 to 0 (not bit-mapped):	
0	no command-specific error
1	(undefined)
2	invalid selection
3	passed parameter too large
4	passed parameter too small
5	too few data bytes received
6	transmitter-specific command error
7	in write-protect mode
8-15	command-specific errors (see Table 4 below)
16	access restricted
32	device is busy
64	command not implemented

Second byte:

Bit 7(hex 80)	device malfunction
Bit 6	configuration changed
Bit 5	cold start
Bit 4	
Bit 3	output current fixed
Bit 2	analog output saturated
Bit 1	variable (not primary) out of limits
Bit 0	primary variable out of limits

Hex equivalents are shown, assuming only a single status indication is present.

These codes have different meanings for different commands. The following table lists some of these meanings. Refer to the full HART specification for information on which codes and meanings are used with each command.

TABLE 4-4. Command-Specific Error Codes.

CODE	MEANING
8	Update Failed Update In Progress Set to Nearest Possible Value
9	Applied Process Too High Lower Range Value Too High Not In Fixed Current Mode
10	Applied Process Too Low Lower Range Value Too Low MultiDrop Not Supported
11	In MultiDrop Mode Invalid Transmitter Variable Code Upper Range Value Too High
12	Invalid Unit Code Upper Range Value Too Low
13	Both Range Values Out of Limits
14	Pushed Upper Range Value Over Limit Span Too Small

Index

Technology Partner
A ROCKWELL AUTOMATION PARTNER

Corporate Headquarters
Spectrum Controls Inc.
1705 132nd Avenue NE, Bellevue, WA 98005 USA
Fax: 425-641-9473
Tel: 425-746-9481

Web Site: www.spectrumcontrols.com
E-mail: spectrum@spectrumcontrols.com

SPECTRUM CONTROLS
An Allient Company