Owner's Guide 0300196-02 Rev. D

CONTROLLOGIX™ HART ANALOG MODULES

Catalog Numbers: 1756sc-IF8H, 1756sc-OF8H

SPECTRUM
CONTROLS

Important Notes

1) **PLEASE DOWNLOAD LATEST SAMPLE PROJECT FROM OUR WEBSITE AT (www.spectrumcontrols.com).**

2) Please read all the information in this owner's guide before installing the product.

3) The information in this owner's guide applies to hardware Series A and firmware version 1.0 or later.

4) 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 assumes no responsibility for the accuracy, completeness, or usefulness of the information herein.

Under no circumstances will Spectrum Controls 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 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 warrants that its products are free from defects in material and workmanship under normal use and service, as described in Spectrum Controls literature covering this product, for a period of 1 year. The obligations of Spectrum Controls 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 facility, transportation charges prepaid, and which after examination is determined, to the satisfaction of Spectrum Controls, to be thus defective.

This warranty shall not apply to any such equipment which shall have been repaired or altered except by Spectrum Controls or which shall have been subject to misuse, neglect, or accident. In no case shall the liability of Spectrum Controls 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.

Table of Contents

Configuring RSLogix 5000 For The IF8H and OF8H 25

Configuration, Data, and Status Tags for the 1756sc-IF8H 37

Configuration, Data, and Status Tags for the 1756sc-OF8H 51

Enabling and Using HART on the 1756sc-IF8H and OF8H 67

Programming Examples 99

Preface

Read this preface to familiarize yourself with the rest of the owner's guide. This preface covers:

- who should use this guide
- what this guide covers
- related Allen-Bradley documents
- terms & abbreviations you should know

Who Should Use This Guide

Use this guide if you design, install, program, or maintain a control system that uses Allen-Bradley ControlLogix Controllers.

You should have a basic understanding of ControlLogix products. You should also understand electronic process control and the ladder program instructions required to generate the electronic signals that control your application. If you do not, contact your local Allen-Bradley representative for the proper training before using these products.

What This Guide Covers

This guide covers the 1756sc-IF8H and 1756sc-OF8H analog input and output modules with HART protocol. It contains the information you need to install, wire, use, and maintain these modules. It also provides diagnostic and troubleshooting help should the need arise.

Related Allen-Bradley Documents

Table A lists several Allen-Bradley documents that may help you as you use these products.

Table A. Related Allen-Bradley documents

Allen-Bradley Doc. No.	Title	Publication Number
1756-PA72, -PB72	ControlLogix Power Supply Installation Instructions	1756-5.1
1756-A4, -A7, -A10, -A13, -A17	ControlLogix Chassis Installation Instructions	1756-5.2

1756 Series ControlLogix Module Installation Instructions (Each module has separate document for installation)	1756-5.5, -5.42
1756-L1, Logix5550 Controller User Manual -L1M1, -L1M2	1756-6.5.12
1756-DHRIO ControlLogix Data Highway Plus Communication Interface Module User Manual	1756-6.5.2
1756-ENET ControlLogix Ethernet Communication Interface Module User Manual	1756-6.5.1

To obtain a copy of any of the Allen-Bradley documents listed, contact your local Allen-Bradley distributor.

Terms & Abbreviations You Should Know

You should understand the following terms and abbreviations before using this guide.

A/D - Refers to analog-to-digital conversion. The conversion produces a digital value whose magnitude is proportional to the instantaneous magnitude of an analog input signal.

Attenuation – The reduction in magnitude of a signal as it passes through a system. The opposite of gain.

Channel – Refers to one of eight, small-signal analog input interfaces to the module's terminal block. Each channel is configured for connection to a input device, and has its own configuration and status words.

Chassis – The component in which the I/O resides. The backplane connection is facilitated through a series of connectors that mate to the I/O.

Common mode rejection ratio (CMRR) - The ratio of a device's differential voltage gain to common mode voltage gain. Expressed in dB, CMRR is a comparative measure of a device's ability to reject interference caused by a voltage common to its terminal relative to ground.

Common mode voltage – The voltage difference between the negative terminal and analog common during normal differential operation.

Cut-off frequency - The frequency at which the input signal is attenuated 3 dB by the digital filter. Frequency components of the input signal that

are below the cut-off frequency are passed with under 3 dB of attenuation for low-pass filters.

Channel Update Time -

dB (decibel) – A logarithmic measure of the ratio of two signal levels.

Digital filter - A low-pass mathmatic single order filter applied to the A/D signal. The digital filter provides high-frequency noise rejection.

Effective resolution – The number of bits in the channel data word that do not vary due to noise.

HART - Highway Addressable Remote Transducer

Local System - A control system with I/O chassis within several feet of the processor.

LSB (least significant bit) – The bit that represents the smallest value within a string of bits.

Multiplexer – A switching system that allows several input signals to share a common A/D converter.

Normal mode rejection (differential mode rejection) – A logarithmic measure, in dB, of a device's ability to reject noise signals between or among circuit signal conductors, but not between the equipment grounding conductor or signal reference structure and the signal conductors.

Module update time – The amount of time that one data acquisition cycle takes place and it reported to the PLC processor.

Remote system - A control system where the chassis can be located several thousand feet from the processor chassis. Chassis communication is via the 1756-CNB or 1756-ENET Adapter.

Resolution – The smallest detectable change in a measurement, typically expressed in engineering units (e.g. 0.15 °C) or as a number of bits. For example, a 12-bit system has 4096 possible output states. It can therefore measure 1 part in 4096. See also effective resolution.

Sampling time - The time required by the A/D converter to sample an input channel.

Step response time – The time required for the A/D signal to reach 95% of its expected, final value, given a full-scale step change in the output data word.

Tags - Identifiers for configuration, data, and status information found within the module. Tags allow the user to modify specific module attributes and view data and status.

Unconnected Message - A ControlNet-based message that is sent without intervention with the PLC controller.

Update time – The time for the module to sample and convert a channel input signal and make the resulting value available to the ControlLogix processor.

Chapter 1

Module Overview

This chapter describes the functionality of the modules and explains how the ControlLogix controller reads/writes analog data from the modules and how HART data is derived from the modules. Read this chapter to familiarize yourself further with your analog module. This chapter covers:

- general description and hardware features
- an overview of system and module operation

General Description

This module is designed exclusively for use in the Allen-Bradley ControlLogix 1756 I/O rack systems. The HART input module, 1756sc-IF8H, stores digitally converted volt (V) and milliamp (mA) analog data and HART field instrument data in its image table for retrieval by all ControlLogix processors. The HART output module, 1756sc-OF8H, produces voltage or current output for control and maintains a dialog with HART-compatible field instruments

Following is a list of features available on the IF8H and OF8H modules that allow their use in a wide variety of applications.

· Removal and insertion under power (RIUP) - a system feature that allows you to remove and insert modules while chassis power is applied

· Producer/consumer communications - an intelligent data exchange between modules and other system devices in which each module produces data without having been polled

· Rolling time stamp of data - 15 bit module-specific rolling timestamp with millisecond resolution which indicates when data was sampled/applied. This timestamp may be used to calculate the interval between channel updates.

· System timestamp of data - 64 bit system clock places a timestamp on the transfer of data between the module and its owner controller within the local chassis

· IEEE 32 bit floating point format

· On-Board Features, such as custom User Scaling, Process Alarms, Rate Alarms, Digital Filtering, and Under/Overrange Detection

· User Calibration - analog I/O modules may be calibrated by the user to accommodate application related errors.

· Class I/Division 2, UL, CSA, CE, and FM Agency Certification

Input Ranges

The following tables provide compatibility information on the supported millivolt and voltage input and output types supported by the modules.

Table 1.1 1756sc-IF8H Input Ranges

0 to 5 V	(0 to +5.125 V)
0 to 10 V	(0 to +10.25 V)
-10 to +10 V	(-10.25 to +10.25 V)
4 to 20 mA	(3.42 to +20.58 mA)
0 to 20 mA	(0 to +20.58 mA)

Table 1.2 1756sc-OF8H Output Ranges

-10 to +10 V	(-10.4 to +10.4 V)
0 to 20 mA	(0 to +21.5 mA)

Eight channels are individually configurable for voltage or current input/output types. Each channel provides wire-off input, over-range, and under-range detection and indication, when enabled.

Hardware Features

The modules fit into any single slot within a ControlLogix modular system. The modules use a unique generic profile which may be configured using your RSLogix 5000 programming software.

The modules utilize a removable terminal block, that provides connections for the eight channels. The module is configured through RSLogix 5000 software, defining current or voltage inputs/outputs.

Table 1.3 Hardware Features

Hardware	Function
OK LED	Displays communication and fault status
Cal LED	Displays a calibration related condition
Side Label (Nameplate)	Provides module information
Removable Terminal Block	Electrical connection to devices
Door Label	Permits easy terminal identification
Self Locking Tabs	Secure module in chassis slot
Terminal Block Switch	Locks the RTB to the module.

Diagnostic LEDs

The modules contain diagnostic LEDs that help you identify the source of problems that may occur during power-up or during normal operation. Power-up and diagnostics are explained in Chapter 9, *Testing Your Module*.

System Overview

The modules communicate with the ControlLogix processor and receive +5 Vdc and +24 Vdc power from the system power supply through the backplane interface. You may install as many modules in the system as the power supply can support.

The 1756sc-IF8H has 8 channels that can receive voltage and current signals from volt or milliamp devices. When configured for volt or milliamp analog inputs, the module converts the analog values directly into floating point values. For those input types, the module assumes that the input signal is linear prior to input into the module.

The 1756sc-OF8H has 8 channels that can output volt or millivolt signals to drive field sensors.

Both modules support HART communication. Communication is independent of the analog acquisition and control phase.

System Operation

At power-up, the modules check internal circuits, memory, and basic functions. During this time the Cal LED remains on. If the module does not find any faults, it turns off the Cal LED.

After completing power-up checks, the modules wait for a connection to an owner controller then valid channel configuration data from your ladder logic program. After channel configuration data is transferred, and one or more channels are enabled, the module channels are available to your ladder program.

Each time the 1756sc-IF8H reads an input channel, the module tests that data for a fault, i.e. over-range, or under-range condition. If it detects an open-circuit (wire off), over-range or under-range condition, the module sets a unique bit in the status tags.

Module Operation

The 1756sc-IF8H module's input circuitry consists of eight single-ended analog inputs, multiplexed to two A/D converters. The A/D converters read the analog input signals and convert them to floating point values.

The 1756sc-OF8H has 8 channels that are routed to two quad D/A converters which output the control signal.

HART communication is independent of the analog operation. Each of the eight channels are multiplexed to one modem.

Compatibility with Current Devices and Cables

The modules are compatible with a variety of voltage and current devices with an input or output 0-5V, 0-10V, ±10V, 0-20mA, and 4-20mA.

To minimize interference from radiated electrical noise, we recommend twisted-pair and highly shielded cables such as the following:

Table 1.4 Recommendations to minimize interference from radiated electrical noise

For This Type of Device	We Recommend This Cable (or equivalent)
V, mA devices	Belden 8761, shielded, twisted-pair

This page is intentionally left blank.

This page is intentionally left blank.

Chapter 2

Installing and Wiring Your Module

Read this chapter to install and wire your module. This chapter covers:

- avoiding electrostatic damage
- determining power requirements
- installing the module
- wiring signal cables to the module's terminal block

Electrostatic Damage

Electrostatic discharge can damage semiconductor devices inside this module if you touch backplane connector pins. Guard against electrostatic damage by observing the following precautions:

Power Requirements

The module receives its power through the ControlLogix chassis backplane from the fixed or modular +5 VDC and +24 VDC chassis power supply. The maximum current drawn by the module is shown in the table below.

Table 2.1 Maximum current drawn by the module

Module	5VDC Amps	24VDC Amps
1756sc-IF8H	0.300	0.070
1756sc-OF8H	0.200	0.230

Using your module in the ControlLogix System

Place your module in any slot of a ControlLogix chassis or modular expansion chassis.

An analog I/O module translates an analog signal into or from a corresponding digital representation which controllers can easily operate on for control purposes.

A ControlLogix I/O module mounts in a ControlLogix chassis and uses a Removable Terminal Block (RTB) to connect all field-side wiring.

Before you install and use your module you should have already:

· installed and grounded a 1756 chassis and power supply.

· ordered and received an RTB for your application.

Important: RTBs are not included with your module purchase.

Specify Allen Bradley Part Number:

1756sc-IF8H -	1756-TBCH - 36 position screw terminals 1756-TBS6H - 36 position press terminals 1492-AIFM8-3 1492-ACABLE-TB
1756sc-OF8H -	1756-TBNH - 20 position screw terminals 1756-TBSH - 20 position press terminals 1492-AIFM8-3 1492-ACABLE-WB

Module Installation and Removal

When installing the module in a chassis, it is not necessary to remove the terminal blocks from the module. However, if the terminal blocks are removed, use the write-on label located on the side of the terminal blocks to identify the module location and type.

Preventing Electrostatic Discharge

This module is sensitive to electrostatic discharge.

ATTENTION: Electrostatic discharge can damage integrated circuits or semiconductors if you touch backplane connector pins. 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 backplane connector or connector pins
· Do not touch circuit components inside the module
· If available, use a static-safe work station
· When not in use, keep the module in its static-shield box

Removal and Insertion Under Power

These modules are designed to be installed or removed while chassis power is applied.

ATTENTION: When you insert or remove a module while backplane power is 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 or loss of process control.

· causing an explosion in a hazardous environment.

Repeated electrical arcing causes excessive wear to contacts on both the module and its mating connectors. Worn contacts may create electrical resistance that can affect module operation.

Compliance to European Union Directives

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

EMC Directive

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

EN 61010-1 and EN 61131-2, EN61000-6-2:2001, EN61000-6-4:2001 EN61010-1:2001

This product is intended for use in an industrial environment.

Low Voltage Directive

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

For specific information required by , EN61131-2:1994 + A11:1996 + A12:2000, see the appropriate sections in this publication, as well as the following Allen-Bradley publications:

· Industrial Automation Wiring and Grounding Guidelines For Noise Immunity, publication 1770-4.1

· Automation Systems Catalog, publication B111

This equipment is classified as open equipment and must be installed (mounted) in an enclosure during operation as a means of providing safety protection.

CAUTION

POSSIBLE EQUIPMENT OPERATION

ATTENTION: **The module is designed to support Removal and Insertion Under Power (RIUP). However, when you remove or insert an RTB with field-side power applied,** unintended machine motion or loss of process control can occur**. Exercise extreme caution when using this feature.**

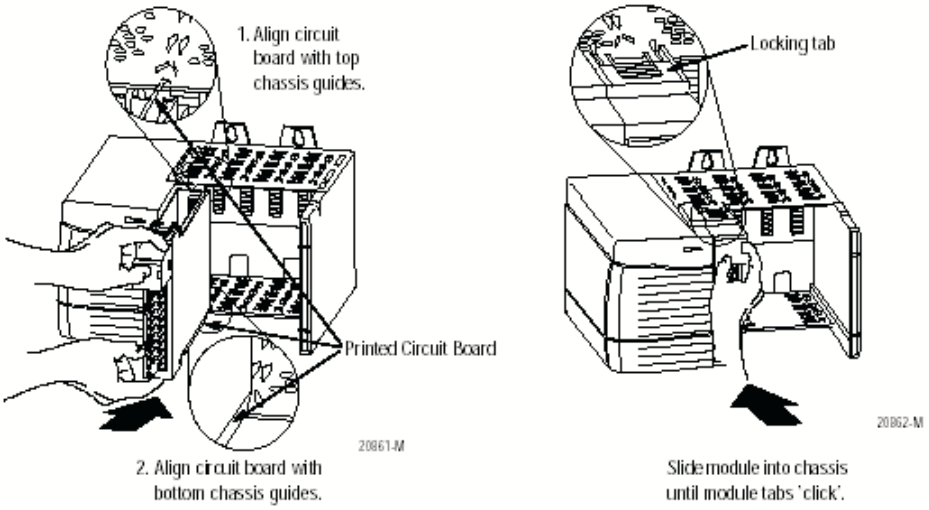
WARNING

These modules are to be used only with the Allen-Bradley 1756 ControlLogix System.

To insert your module into the rack, follow these steps:

1. Align the circuit board of your module with the card guides at the top and bottom of the chassis.

Figure 2.1 (Module insertion into a rack)



2. Key the RTB in positions that correspond to unkeyed module positions. Insert the wedge-shaped tab on the RTB with the rounded edge first. Push the tab onto the RTB until it stops.

Keying the Removable Terminal Block

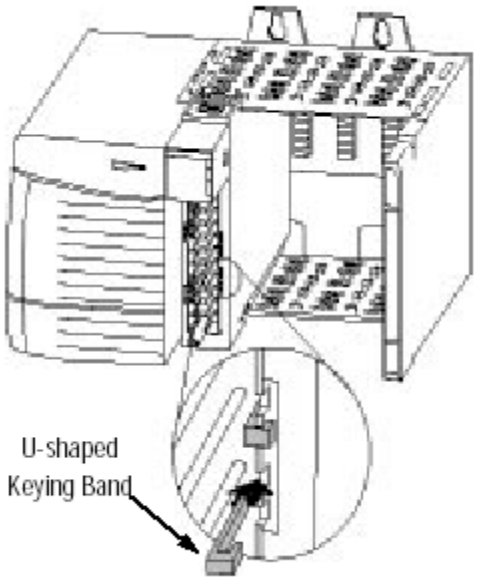
Key the RTB to prevent inadvertently connecting the incorrect RTB to your module.

When the RTB mounts onto the module, keying positions will match up. For example, if you place a U-shaped keying band in position #4 on the module, you cannot place a wedge-shaped tab in #4 on the RTB or your RTB will not mount on the module.

We recommend that you use a unique keying pattern for each slot in the chassis.

1. Insert the U-shaped band with the longer side near the terminals. Push the band onto the module until it snaps into place.

Figure 2.2 (Terminal block diagram with keying)



Wiring Your Module

Follow these guidelines to wire your input signal cables:

- Power, input, and output (I/O) wiring must be in accordance with Class 1, Division 2 wiring methods [Article 501-4(b) of the National Electrical Code, NFPA 70] and in accordance with the authority having jurisdiction.
- Peripheral equipment must be suitable for the location in which it is used.
- Route the field wiring away from any other wiring and as far as possible from sources of electrical noise, such as motors, transformers, contactors, and ac devices. As a general rule, allow at least 6 in. (about 15.2 cm) of separation for every 120 V of power.
- Routing the field wiring in a grounded conduit can reduce electrical noise further.
- If the field wiring must cross ac or power cables, ensure that they cross at right angles.
- To limit the pickup of electrical noise keep signal wires as far from power and load lines as possible.
- For improved immunity to electrical noise, use Belden 8761 (shielded, twisted pair) or equivalent wire for millivolt sensors

- Ground the shield drain wire at only one end of the cable. The preferred location is at the shield connections at the ControlLogix chassis. (Refer to IEEE Std. 518, Section 6.4.2.7 or contact your sensor manufacturer for additional details.)
- Keep all unshielded wires as short as possible.
- To limit overall cable impedance, keep input cables as short as possible. Locate your I/O chassis as near to the sensors as your application will permit.
- Tighten screw terminals with care. Excessive tightening can strip a screw.
- Follow system grounding and wiring guidelines found in your ControlLogix Installation and Operation Manual.

Preparing and Wiring the Cables

To prepare and connect cable leads and drain wires, follow these steps:

1. At each end of the cable, strip some casing to expose individual wires.
2. Trim signal wires to 5-inch lengths beyond the cable casing. Strip about 3/16 inch (4.76 mm) of insulation to expose the ends of the wires.
3. At the module-end of the cables (see figure above):

   - extract the drain wire and signal wires

   - remove the foil shield

   - bundle the input cables with a cable strap
4. Connect pairs of drain wires together, Channels 0 and 1, Channels 2 and 3, Channels 4 and 5, Channels 6 and 7. Keep drain wires as short as possible.
5. Connect the drain wires to the grounding lug on the PLC chassis.
6. Connect the signal wires of each channel to the terminal block. Important: Only after verifying that your connections are correct for each channel, trim the lengths to keep them short. Avoid cutting leads too short.
7. At the source-end of cables from voltage devices:

   - remove the drain wire and foil shield

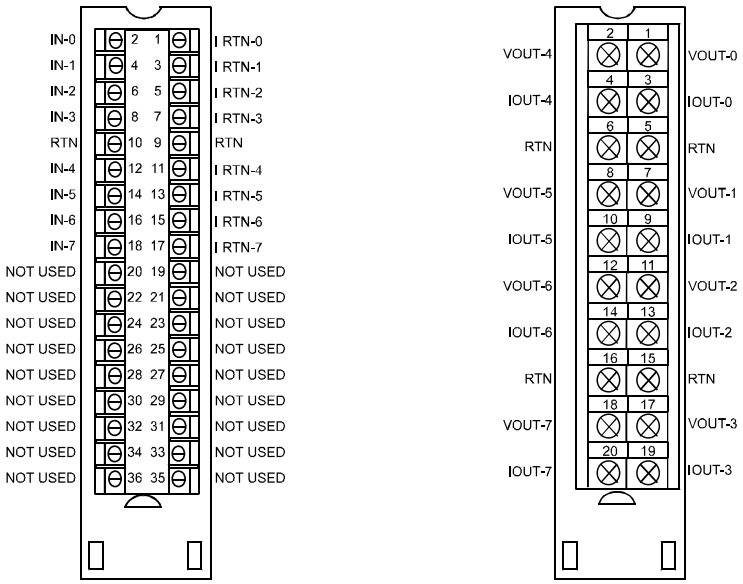
   - apply shrink wrap as an option

- connect to devices keeping the leads short

Important: If noise persists, try grounding the opposite end of the cable, instead (Ground one end only.)

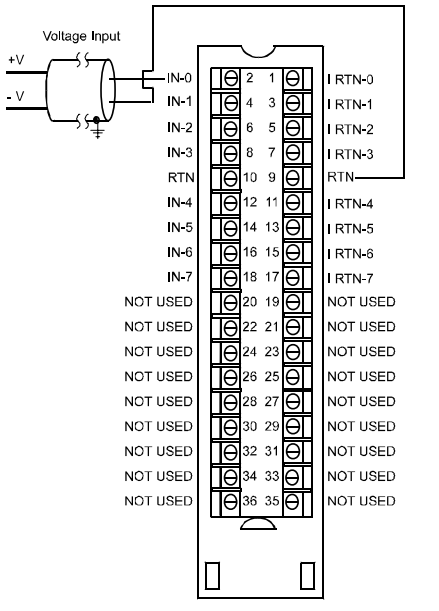
Terminal Block Layout

The following figure shows the general terminal block layout. The input signal type will determine which pins are used.



Wiring Inputs to the IF8H Module

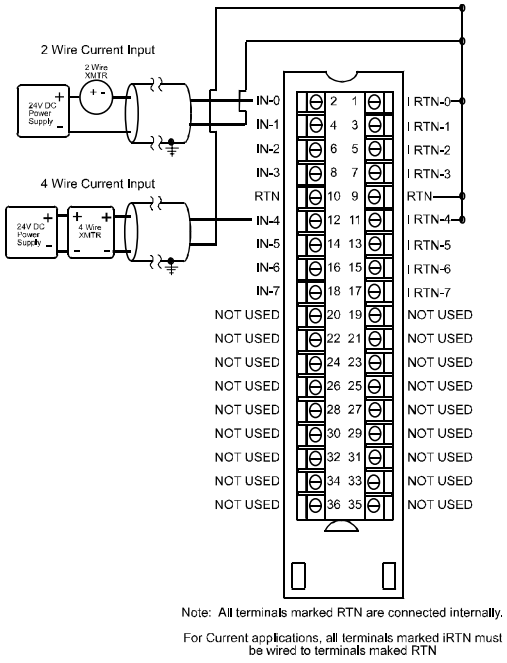
Voltage Inputs - Voltage inputs use the terminal block pins labelled IN-# and RTN



Note: All terminals marked RTN are connected internally.

Current Inputs - Current inputs use the terminal block pins labelled IN-#, i RTN-# and RTN Note that HART communication is only active with current inputs.

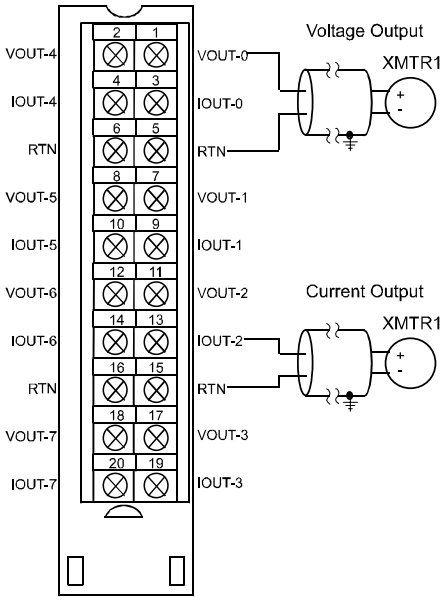
Attention! 4 wire devices may be more susceptible to electrical nose and ground loops when used with a single ended analog module.



Wiring Outputs to the OF8H Module

The OF8H module supports voltage and current outputs.

Voltage Outputs - Voltage outputs use the terminal block pins labelled VOUT-# and RTN



Current Outputs - Current outputs use the terminal block pins labelled IOUT-# and RTN

Note: HART communication is only active with current outputs.

This page intensionally left blank.

Operation Within the ControlLogix System

This chapter describes how the 1756sc-IF8H and 1756sc-OF8H analog HART modules work within the ControlLogix system. This chapter covers:

- Ownership and connections to the module
- Direct connections
- Listen only mode
- Configuration changes with multiple owners.

Ownership and Connections

Every I/O module in the ControlLogix system must be owned by a Logix5550 Controller to be useful. This owner-controller stores configuration data for every module that it owns and can be local or remote in regard to the I/O module's position. The owner sends the I/O module configuration data to define the module's behavior and begin operation within the control system. Each ControlLogix I/O module must continuously maintain communication with its owner to operate normally.

Typically, each module in the system will have only 1 owner. Input modules can have more than 1 owner. Output modules, however, are limited to a single owner.

Using RSNetWorx and RSLogix 5000

The I/O configuration portion of RSLogix5000 generates the configuration data for each I/O module in the control system, whether the module is located in a local or remote chassis. A remote chassis, also known as networked, contains the I/O module but not the module's owner controller. Configuration data is transferred to the controller during the program download and subsequently transferred to the appropriate I/O modules. I/O modules in the same chassis as the controller are ready to run as soon as the configuration data has been downloaded. You must run RSNetWorx to enable I/O modules in the networked chassis.

Running RSNetWorx transfers configuration data to networked modules and establishes a Network Update Time (NUT) for ControlNet that is compliant with the desired communications options specified for each module during configuration. If you are not using I/O modules in a networked chassis, running RSNetWorx is not necessary. However, anytime a controller references an I/O module in a networked chassis,

RSNetWorx must be run to configure ControlNet. Follow these general guidelines when configuring I/O modules:

1. Configure all I/O modules for a given controller using RSLogix 5000 and download that information to the controller.

2. If the I/O configuration data references a module in a remote chassis, run RSNetWorx.

Important: RSNetWorx **must** be run whenever a new module is added to a networked chassis. When a module is permanently removed from a remote chassis, we recommend that RSNetWorx be run to optimize the allocation of network bandwidth.

Direct Connections

A **direct connection** is a real-time data transfer link between the controller and the device that occupies the slot that the configuration data references. When module configuration data is downloaded to an owner-controller, the controller attempts to establish a direct connection to each of the modules referenced by the data.

If a controller has configuration data referencing a slot in the control system, the controller periodically checks for the presence of a device there. When a device's presence is detected, the controller automatically sends the configuration data. If the data is appropriate to the module found in the slot, a connection is made and operation begins. If the configuration data is not appropriate, the data is rejected and an error message displays in the software. In this case, the configuration data can be inappropriate for any of a number of reasons.

The controller maintains and monitors its connection with a module. Any break in the connection, such as removal of the module from the chassis while under power, causes the controller to set fault status bits in the data area associated with the module. The RSLogix 5000 software may monitor this data area to announce the modules' failures.

Module Operation

In traditional I/O systems, controllers poll input modules to obtain their input status. Analog input modules in the ControlLogix system are not polled by a controller once a connection is established. The modules multicast their data periodically. Multicast frequency depends on the options chosen during configuration and where in the control system that input module physically resides. An input module's communication, or multicasting, behavior varies depending upon whether it operates in the local chassis or in a remote chassis. The following sections detail the differences in data transfers between these set-ups.

Modules in a Local Chassis

When a module resides in the same chassis as the owner controller, the following two configuration parameters will affect how and when the input module multicasts data:

· Real Time Sample (RTS) configured via Real Time Sample tag.

· Requested Packet Interval (RPI) configured via I/O module properties.

Real Time Sample (RTS)

This configurable parameter instructs the module to perform the following operations:

1. scan all of its input channels and store the data into on-board memory

2. multicast the updated channel data (as well as other status data) to the backplane of the local chassis

Requested Packet Interval (RPI)

This configurable parameter also instructs the module to multicast its channel and status data to the local chassis backplane.

The RPI instructs the module to multicast the **current contents** of its on-board memory when the RPI expires, (i.e. the module does not update its channels prior to the multicast).

Important: The RPI value is set during the initial module configuration using RSLogix 5000.

It is important to note that the module will reset the RPI timer each time an RTS is performed. This operation dictates how and when the owner controller in the local chassis will receive updated channel data, depending on the values given to these parameters. If the RTS value is less than or equal to the RPI, each multicast of data from the module will have updated channel information. In effect, the module is only multicasting at the RTS rate.

If the RTS value is greater than the RPI, the module will multicast at both the RTS rate and the RPI rate. Their respective values will dictate how often the owner controller will receive data and how many multicasts from the module contain updated channel data. Note: Even though data may be transferred at the RPI rate, the data will be identical to the previous RTS data transfer. HART data can change. Setting the RPI < 100 msec will allow you to see all changes.

Modules in a Remote Chassis

If an input module resides in a networked chassis, the role of the RPI and the module's RTS behavior change slightly with respect to getting data to the owner. The RPI and RTS intervals still define when the module will multicast data **within its own chassis** (as described in the previous section), but only the value of the RPI determines how often the owner controller will receive it over the network.

When an RPI value is specified for an input module in a remote chassis, in addition to instructing the module to multicast data within its own chassis, the RPI also "reserves" a spot in the stream of data flowing across the ControlNet network.

The timing of this "reserved" spot may or may not coincide with the exact value of the RPI, but the control system will guarantee that the owner controller will receive data **at least as often** as the specified RPI.

The "reserved" spot on the network and the module's RTS are asynchronous to each other. This means there are Best and Worst Case scenarios as to when the owner controller will receive updated channel data from the module in a networked chassis.

Best Case RTS Scenario

In the Best Case scenario, the module performs an RTS multicast with updated channel data just before the "reserved" network slot is made available. In this case, the remotely located owner receives the data almost immediately.

Worst Case RTS Scenario

In the Worst Case scenario, the module performs an RTS multicast just after the "reserved" network slot has passed. In this case, the owner-controller will not receive data until the next scheduled network slot.

Because it is the RPI and NOT the RTS which dictates when the module's data will be sent over the network, we recommend the RPI value be set LESS THAN OR EQUAL TO the RTS to make sure that updated channel data is received by the owner controller with each receipt of data.

Listen-Only Mode

Any controller in the system can **listen** to the data from any I/O module (e.g. input data or "echoed" output data) even if the controller does not own the module (i.e. it does not have to hold the module's configuration data to listen to the module).

The "listen only" mode is set during the I/O configuration process.

Choosing a 'Listen-Only' mode option allows the controller and module to establish communications without the controller sending

any configuration data. In this instance, another controller owns the module being listened to.

Important: Controllers using the Listen-Only mode continue to receive data multicast from the I/O module as long as a connection between an owner and I/O module is maintained. If the connection between all owners and the module is broken, the module stops multicasting data and connections to all 'Listening controllers' are also broken.

Multiple Owners of Input Modules

Because 'Listening controllers' lose their connections to modules when communications with the owner stop, the ControlLogix system will allow you to define more than one owner for input modules.

Important: Only input modules can have multiple owners. If multiple owners are connected to the same input module, they **must maintain identical configuration** for that module.

In the example below, Controller A and Controller B have both been configured to be the owner of the input module.

When the controllers begin downloading configuration data, both try to establish a connection with the input module. Whichever controller's data arrives first establishes a connection. When the second controller's data arrives, the module compares it to its current configuration data (the data received and accepted from the first controller).

If the configuration data sent by the second controller matches the configuration data sent by the first controller the connection is also accepted. If any parameter of the second configuration data is different from the first, the module rejects the connection and the user is informed by an error in the software.

The advantage of multiple owners over a 'Listen-only' connection is that now either of the controllers can lose the connection to the module and the module will continue to operate and multicast data to the system because of the connection maintained by the other owner controller.

Note: The previous discussion of multiple owners assumes the configuration tag ".configrevnumber" is set to 1. Operation differs is the tag is set to 0. Refer to Chapter 5 for descriptions of this tag's settings.

Configuration Changes in an Input Module with Multiple Owners

You must be careful when changing an input module's configuration data in a multiple owner scenario. When the configuration data is changed in one of the owners, for example, Controller A, and sent to the module, that configuration data is accepted as the new configuration for the module. Controller B will continue to listen, unaware that any changes have been made in the module's behavior.

Important: When changing configuration for a module with multiple owners, we recommend the connection be inhibited. To prevent other owners from receiving potentially erroneous data, as described above, the following steps **must be followed** when changing a module's configuration in a multiple owner scenario when on-line:

1. For each owner controller, inhibit the controller's connection to the module in the software on the I/O Module Connection tab.

2. Make the appropriate configuration data changes in the software.

3. Repeat steps 1 and 2 for all owner controllers, making the **exact same changes** in all controllers.

4. Uncheck the Inhibit box in each owner's configuration to reconnect each module.

Chapter 4

Configuring RSLogix 5000 For The IF8H and OF8H

This chapter explains how to incorporate your module into the ControlLogix system. It also covers a brief overview of the configuration, input and output (OF8H only) tags Topics discussed include:

- Adding your module to a RSLogix 5000 project
- Configuration tags overview
- Input tags overview
- Output tags overview

Module Installation

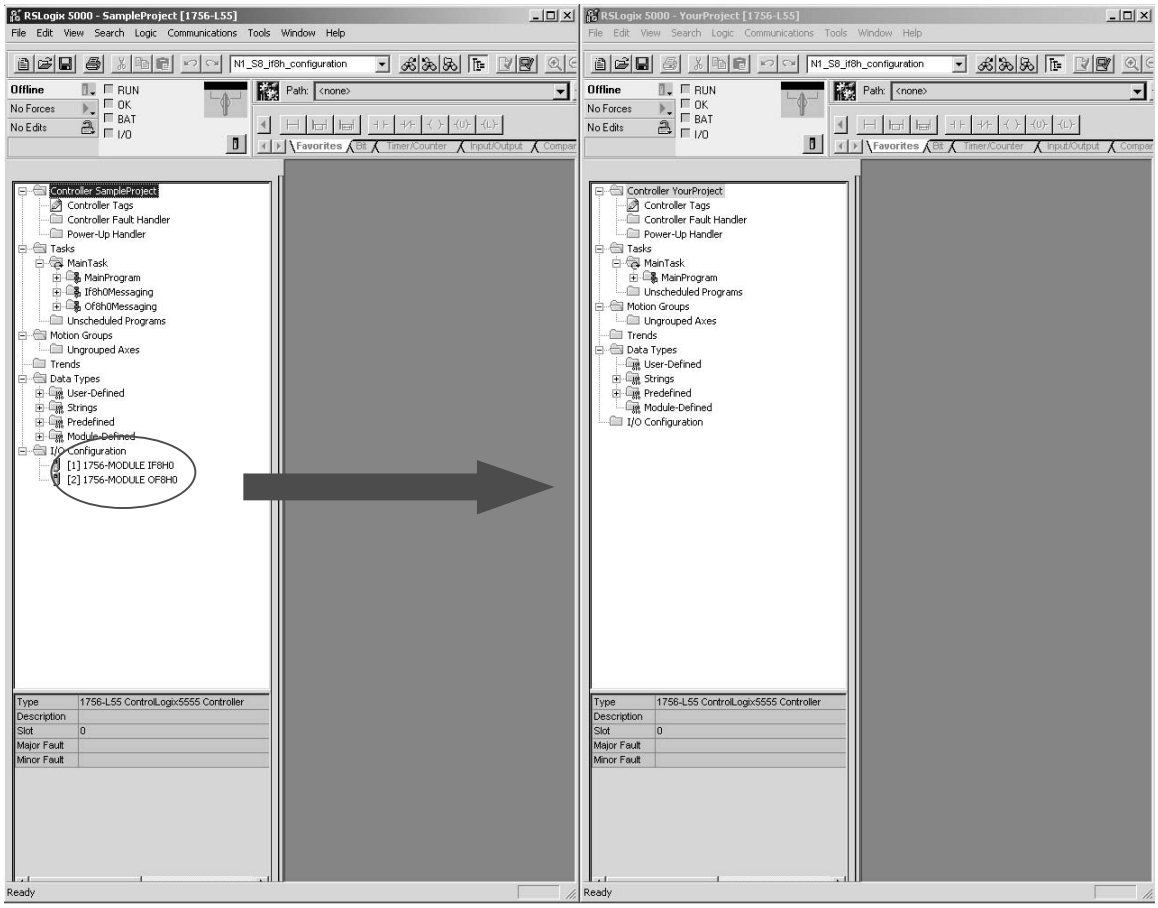
The process of incorporating your HART module into the ControlLogix system is similar to the process needed to add an Allen-Bradley module. You will use your RSLogix 5000 programming software to install and configure your HART module. The module is not currently in the RSLogix 5000 I/O pick list, so you will need to copy and paste information from a sample project that can be obtained from our website at (www.spectrumcontrols.com). You may also choose to build onto the sample project itself. The sample project contains the module profile, configuration tags, input tags, and ladder samples needed to configure each HART module.

Adding Your Module to a Project

The module has a unique set of tag definitions which are used to configure specific features. Chapter 5 and 6, *Channel Configuration, Data, and Status*, gives you detailed information about available configuration settings and status information. These values are set using your programming

software and ladder logic. Before you can use these features you must first include the module into the project.

1. Download and open the sample project from our website at www.spectrumcontrols.com It contains information for the IF8H and OF8H. Open your project. Drag and drop the IF8H or OF8H module into the I/O configuration section of your project.



a) Open the sample project.

b) Open your new project.

c) Click once on the IF8H or OF8H in the IO configurator.

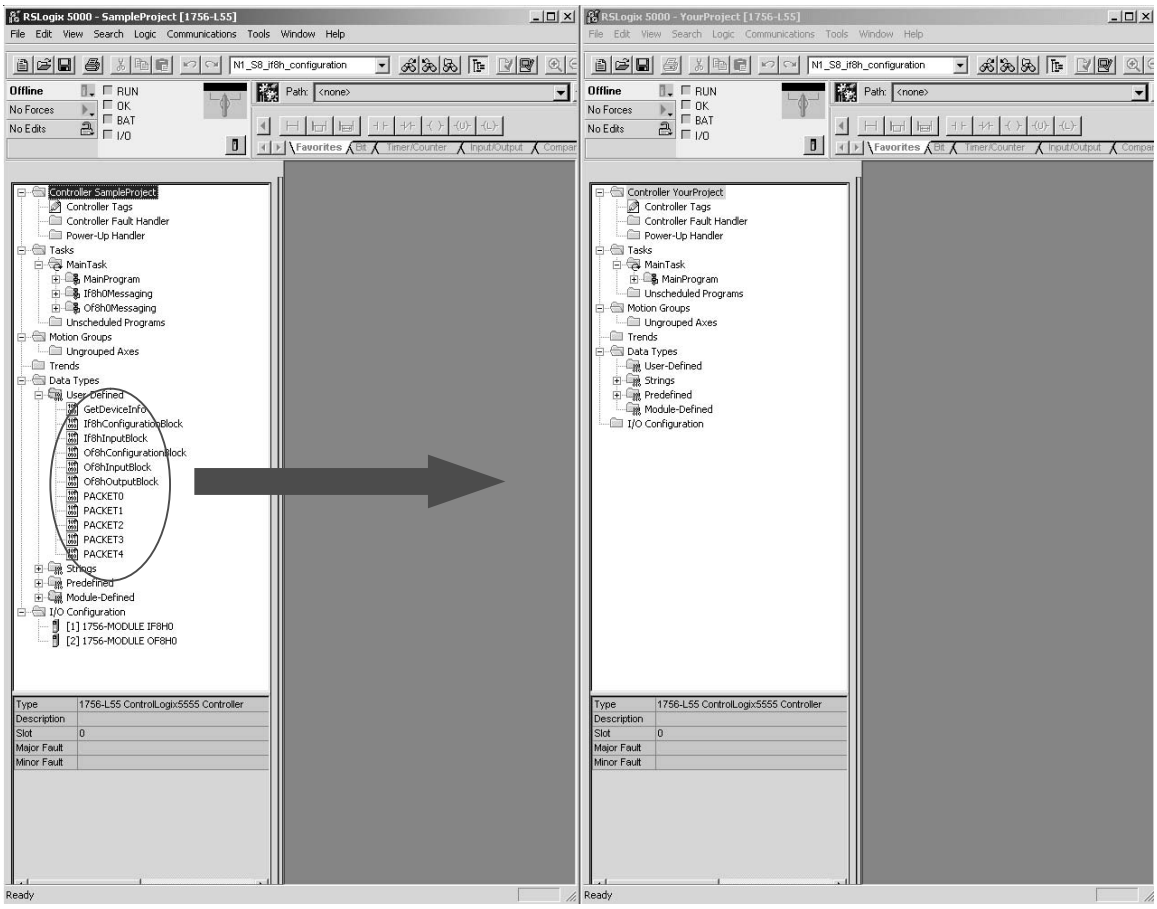
d) Drag and drop it into the I/O Configurator section of your project.

Note: You may need to change the slot number of the module after pasting it into your project.

Note: If only one of the HART modules is to be utilized, copy the profile, tags and ladder for that module only.

See Appendix C for more details regarding module profile settings.

2. Drag and drop the IF8H or OF8H user-defined data types from the sample project into your project.



There are eleven user defined data types that need to be moved.

- GetDeviceInfo
- If8hConfigurationBlock
- Of8hConfigurationBlock
- If8hInputBlock
- Of8hInputBlock
- Of8hOutputBlock
- Packet0
- Packet1
- Packet2
- Packet3
- Packet4

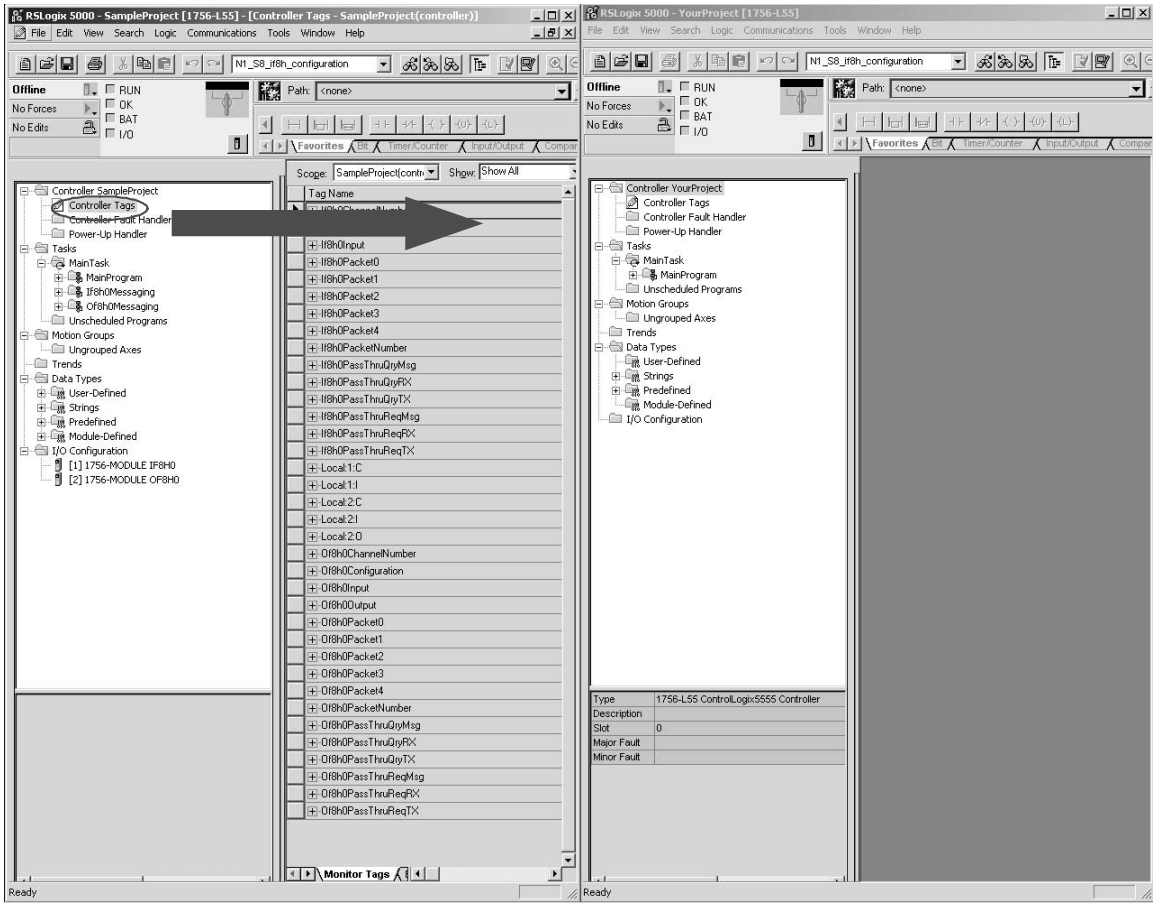
a) Click on the data type

b) Drag it into your new project.

c) Continue to drag and drop the data types until all have been moved.

Note: These can only be moved one at a time.

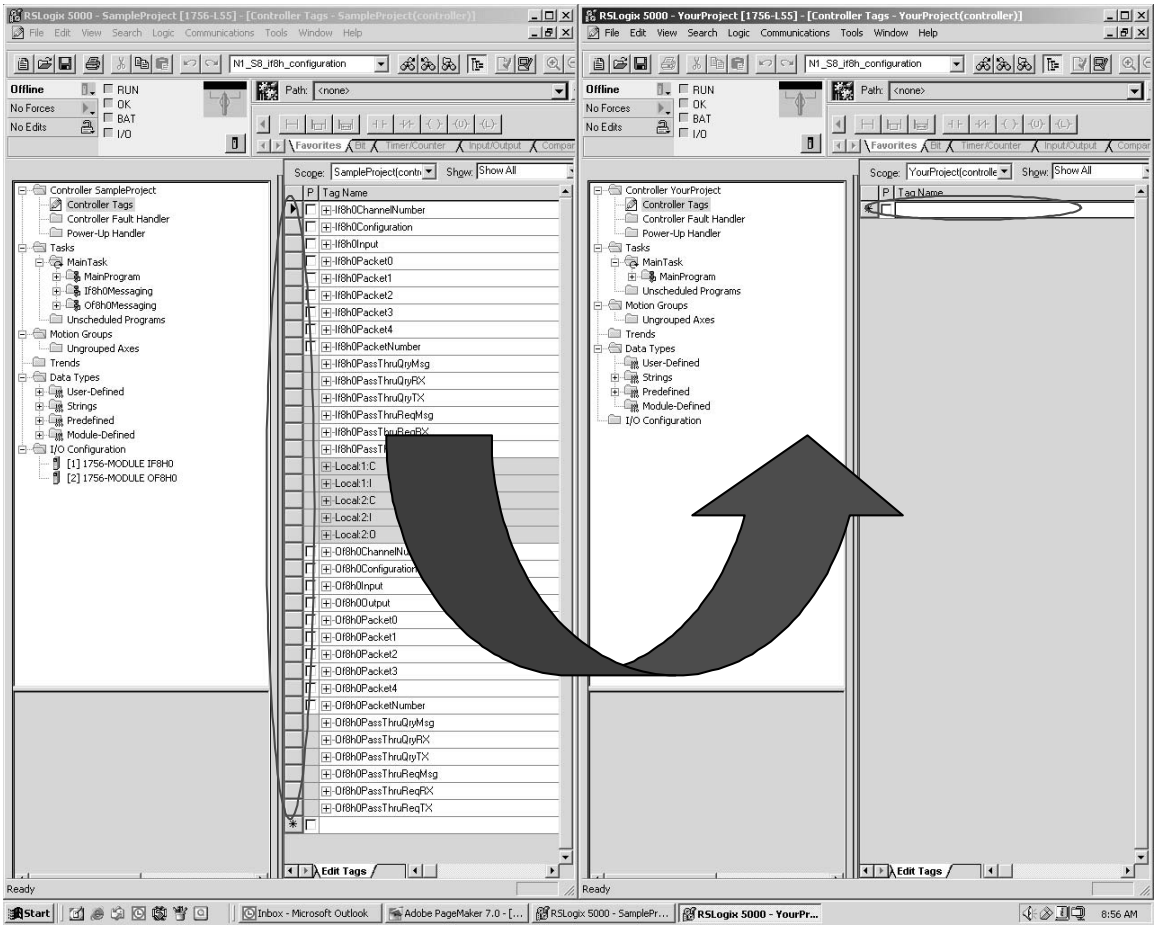
3. Drag and drop the controller tags from the sample project into your project.



a) Right click on the Controller Tags item of the sample project and select edit.

b) Right click on the Controller Tags item of your project and select edit.

c) Scroll down to the Controller tags of the sample project and select all the tags by highlighting them. Be sure to select the tags by using the gray buttons to the left of the tag name. See figure below.



d) Paste the tags into your project. Be sure to paste the tags in the empty field marked with an asterisk. Refer to the figure above.

Note: If you are sending HART messages to the module, you will need to copy the If8h0Messaging program and or the Of8h0Messaging program. You will also need to copy the associated program tags for each routine. Use the procedure in step 3 for copying the program tags to your project.

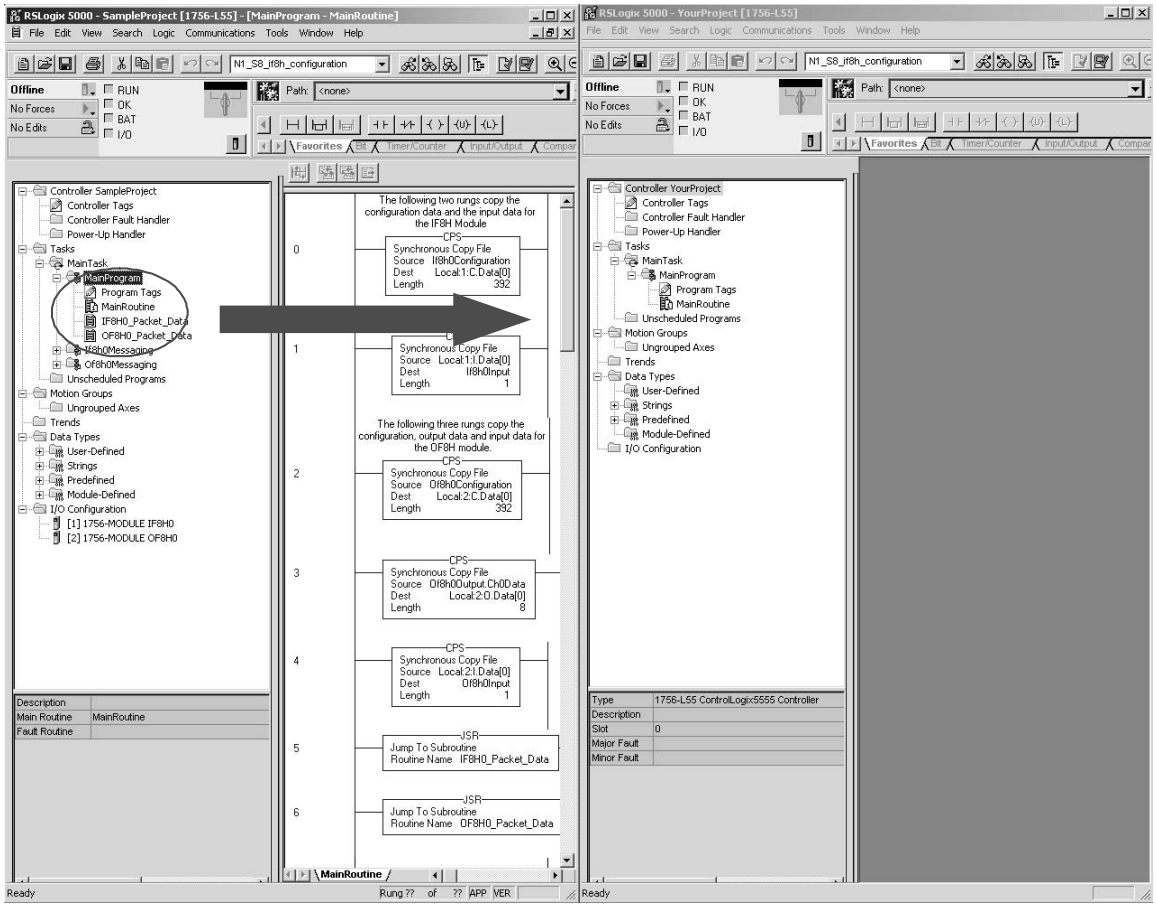
Note: If8h0Configuration / If8h0Input and Of8h0Configuration / Of8h0Input contain the configuration, data and status tags for the modules. The other tags are used for performing various functions to your module via ladder logic.

Note: Certain tag names include a zero succeeding the catalog number; for example, If8h0Configuration contains a zero. The zero is used to identify the module if there's more than one IF8H module in the system. This number could also be used to imply the slot number of the module. The user can omit this number or change the tag name if need be.

Note: Be sure all tags are displayed before moving them. Select Display All from the Edit drop down window.

Note: The “Local:e:I” and “Local:e:C” tags are not copied.

4. Create a new ladder logic routine in your project.



a) In your project, double click on the MainRoutine.

b) Double click on the MainRoutine item in the sample project to display the ladder logic.

c) Left mouse inside the MainProgram ladder logic in the sample project and press crtl-A to select all the rungs.

d) Drag and drop these rungs over and add them to the MainRoutine in your project.

Note: You will need to delete the one blank “solid bar” rung either at the top or bottom of the routine which was left over when you double clicked on the empty MainRoutine in your project.

e) Drag and drop the routines IF8H0_Packet_Data and OF8H0_Packet_Data to the MainProgram in your project.

Note: You may choose to omit pasting the OF8H0_Packet_Data routine if you are not using the OF8H module in your system.

f) Now add JSR ladder instructions in your MainRountine which calls the two routines in step e.

Note: RSLogix 5000 will verify the ladder logic sample. You may receive errors regarding invalid tags. You will need to change the slot addressing in the logic to coordinate with the location of the module.

This completes the installation of the module in the system

Configuration Tags Overview

The configuration tags for the IF8H and OF8H that were copied in step 3 are used to change the operation settings, including input type, filter-frequency, scaling, etc. The data contained in these tags are then copied to both the IF8H and OF8H local configuration tags by the ladder in step 4. When using generic profiles, the local module tags that are created are made up of a single dimensional array with only one data type allowed, usually a DINT. Therefore, the process of copying the defined configuration tags to the local configuration tags is required in order to parse the data. Specific information regarding tag settings may be found in Chapter 5.

Note: The local configuration tags (i.e Local:e:C) mentioned above are automatically created when the module was copied from the sample project in step 1.

Note: The defined configuration tags (If8h0Configuration) are copied to the local configuration tags (Local:e:C) by the ladder in step 4.

1756sc-IF8H (If8h0Configuration)

If8h0Configuration	{...}	{...}		If8hConfigur...
If8h0Configuration.Revision	16#00		Hex	SINT
If8h0Configuration.AdcFilter	6		Decimal	SINT
If8h0Configuration.RTS	250		Decimal	INT
If8h0Configuration.Ch0ConfigBits	2#1000_0000		Binary	SINT
If8h0Configuration.Ch0InputRange	4		Decimal	SINT
If8h0Configuration.Ch0DigitalFilter	0		Decimal	INT
If8h0Configuration.Ch0RateAlarmLimit	0.0		Float	REAL
If8h0Configuration.Ch0LowSignal	4.0		Float	REAL
If8h0Configuration.Ch0HighSignal	20.0		Float	REAL
If8h0Configuration.Ch0LowEngineering	4.0		Float	REAL
If8h0Configuration.Ch0HighEngineering	20.0		Float	REAL
If8h0Configuration.Ch0LowAlarm	0.0		Float	REAL
If8h0Configuration.Ch0HighAlarm	0.0		Float	REAL
If8h0Configuration.Ch0LowLowAlarm	0.0		Float	REAL
If8h0Configuration.Ch0HighHighAlarm	0.0		Float	REAL
If8h0Configuration.Ch0AlarmDeadband	0.0		Float	REAL
If8h0Configuration.Ch0CalBias	0.0		Float	REAL
⋮	⋮		⋮	
If8h0Configuration.Ch7ConfigBits	2#1000_0000		Binary	SINT
If8h0Configuration.Ch7InputRange	4		Decimal	SINT
If8h0Configuration.Ch7DigitalFilter	0		Decimal	INT
If8h0Configuration.Ch7RateAlarmLimit	0.0		Float	REAL
If8h0Configuration.Ch7LowSignal	4.0		Float	REAL
If8h0Configuration.Ch7HighSignal	20.0		Float	REAL
If8h0Configuration.Ch7LowEngineering	4.0		Float	REAL
If8h0Configuration.Ch7HighEngineering	20.0		Float	REAL
If8h0Configuration.Ch7LowAlarm	0.0		Float	REAL
If8h0Configuration.Ch7HighAlarm	0.0		Float	REAL
If8h0Configuration.Ch7LowLowAlarm	0.0		Float	REAL
If8h0Configuration.Ch7HighHighAlarm	0.0		Float	REAL
If8h0Configuration.Ch7AlarmDeadband	0.0		Float	REAL
If8h0Configuration.Ch7CalBias	0.0		Float	REAL
If8h0Configuration.HandleTimeout	5		Decimal	INT
If8h0Configuration.ModuleConfigBits	2#0000_0000_0000_0000		Binary	INT

1756sc-OF8H (Of8h0Configuration)

Of8h0Configuration	{...}	{...}		Of8hConfigu...
Of8h0Configuration.ConfigRevision	16#00		Hex	SINT
Of8h0Configuration.ProgToFaultEn	0		Decimal	SINT
Of8h0Configuration.Spare	0		Decimal	INT
Of8h0Configuration.Ch0ConfigBits	2#0000_0010_0000_0000		Binary	INT
Of8h0Configuration.Ch0OutputRange	2		Decimal	INT
Of8h0Configuration.Ch0MaxRampRate	0.0		Float	REAL
Of8h0Configuration.Ch0FaultValue	0.0		Float	REAL
Of8h0Configuration.Ch0IdleValue	0.0		Float	REAL
Of8h0Configuration.Ch0LowSignal	4.0		Float	REAL
Of8h0Configuration.Ch0HighSignal	20.0		Float	REAL
Of8h0Configuration.Ch0LowEngineering	4.0		Float	REAL
Of8h0Configuration.Ch0HighEngineering	20.0		Float	REAL
Of8h0Configuration.Ch0LowClamp	4.0		Float	REAL
Of8h0Configuration.Ch0HighClamp	20.0		Float	REAL
Of8h0Configuration.Ch0CalBias	0.0		Float	REAL
Of8h0Configuration.Ch0Slot0Code	0		Decimal	SINT
Of8h0Configuration.Ch0Slot1Code	0		Decimal	SINT
Of8h0Configuration.Ch0Slot2Code	0		Decimal	SINT
Of8h0Configuration.Ch0Slot3Code	0		Decimal	SINT
⋮	⋮		⋮	
Of8h0Configuration.Ch7ConfigBits	2#0000_0010_0000_0000		Binary	INT
Of8h0Configuration.Ch7OutputRange	2		Decimal	INT
Of8h0Configuration.Ch7MaxRampRate	0.0		Float	REAL
Of8h0Configuration.Ch7FaultValue	0.0		Float	REAL
Of8h0Configuration.Ch7IdleValue	0.0		Float	REAL
Of8h0Configuration.Ch7LowSignal	4.0		Float	REAL
Of8h0Configuration.Ch7HighSignal	20.0		Float	REAL
Of8h0Configuration.Ch7LowEngineering	4.0		Float	REAL
Of8h0Configuration.Ch7HighEngineering	20.0		Float	REAL
Of8h0Configuration.Ch7LowClamp	4.0		Float	REAL
Of8h0Configuration.Ch7HighClamp	20.0		Float	REAL
Of8h0Configuration.Ch7CalBias	0.0		Float	REAL
Of8h0Configuration.Ch7Slot0Code	0		Decimal	SINT
Of8h0Configuration.Ch7Slot1Code	0		Decimal	SINT
Of8h0Configuration.Ch7Slot2Code	0		Decimal	SINT
Of8h0Configuration.Ch7Slot3Code	0		Decimal	SINT
Of8h0Configuration.HandleTimeout	5		Decimal	INT
Of8h0Configuration.ModuleConfigBits	2#1000_0000_0000_0000		Binary	INT

Input Tags Overview

The input tags contain the analog data, status, and HART data. As in the case of the configuration tags, the input tags are also copied to the local input tags for the module. For details regarding the analog data and status refer to Chapter 5. For details regarding the HART data refer to Chapter 7.

Note: The local input tags (i.e. Local:e:I) mentioned above are automatically created when the module was copied from the sample project in step 1.

Note: The defined configuration tags (If8h0Input) are copied to the local input tags (Local:e:I) by the ladder in step 4.

Note: The HART data found in the input tags has not been demultiplexed. In other words, the data is changing dynamically depending on what channel and which HART packet is currently being scanned. Please refer to chapter 7 for more details regarding demultiplexing of the HART data.

1756sc-IF8H (If8h0Input)

⊞ If8h0Configuration	{...}	{...}		If8hConfiguratio...
⊟ If8h0Input	{...}	{...}		If8hInputBlock
⊞ If8h0Input.ModuleStatus	2#1000_0000_0000_0000_1111_1100_1111_1100		Binary	DINT
⊟ If8h0Input.ChanStatus	{...}	{...}	Binary	SINT[8]
⊞ If8h0Input.ChanStatus[0]	2#0000_0000		Binary	SINT
⊞ If8h0Input.ChanStatus[1]	2#0000_0000		Binary	SINT
⊞ If8h0Input.ChanStatus[2]	2#0010_0000		Binary	SINT
⊞ If8h0Input.ChanStatus[3]	2#0010_0000		Binary	SINT
⊞ If8h0Input.ChanStatus[4]	2#0010_0000		Binary	SINT
⊞ If8h0Input.ChanStatus[5]	2#0010_0000		Binary	SINT
⊞ If8h0Input.ChanStatus[6]	2#0010_0000		Binary	SINT
⊞ If8h0Input.ChanStatus[7]	2#0010_0000		Binary	SINT
⊟ If8h0Input.ChanData	{...}	{...}	Float	REAL[8]
If8h0Input.ChanData[0]	7.6927147		Float	REAL
If8h0Input.ChanData[1]	7.579315		Float	REAL
If8h0Input.ChanData[2]	20.58		Float	REAL
If8h0Input.ChanData[3]	20.58		Float	REAL
If8h0Input.ChanData[4]	20.58		Float	REAL
If8h0Input.ChanData[5]	20.58		Float	REAL
If8h0Input.ChanData[6]	20.58		Float	REAL
If8h0Input.ChanData[7]	20.58		Float	REAL
⊞ If8h0Input.CST	{...}	{...}	Hex	DINT[2]
⊞ If8h0Input.TimeStamp	2826		Decimal	INT
⊞ If8h0Input.HartData	{...}	{...}	Decimal	SINT[40]

1756sc-OF8H (Of8hInput)

Of8h0Input	{...}	{...}		Of8hInputBlock
Of8h0Input.ModuleStatus	2#1000_0000_0000_0000_0000_0000_1111_1111		Binary	DINT
Of8h0Input.ChanStatus	{...}	{...}	Binary	SINT[8]
Of8h0Input.ChanStatus[0]	2#0000_0010		Binary	SINT
Of8h0Input.ChanStatus[1]	2#0000_0010		Binary	SINT
Of8h0Input.ChanStatus[2]	2#0000_0010		Binary	SINT
Of8h0Input.ChanStatus[3]	2#0000_0010		Binary	SINT
Of8h0Input.ChanStatus[4]	2#0000_0010		Binary	SINT
Of8h0Input.ChanStatus[5]	2#0000_0010		Binary	SINT
Of8h0Input.ChanStatus[6]	2#0000_0010		Binary	SINT
Of8h0Input.ChanStatus[7]	2#0000_0010		Binary	SINT
Of8h0Input.ChanData	{...}	{...}	Float	REAL[8]
Of8h0Input.ChanData[0]	4.0		Float	REAL
Of8h0Input.ChanData[1]	4.0		Float	REAL
Of8h0Input.ChanData[2]	4.0		Float	REAL
Of8h0Input.ChanData[3]	4.0		Float	REAL
Of8h0Input.ChanData[4]	4.0		Float	REAL
Of8h0Input.ChanData[5]	4.0		Float	REAL
Of8h0Input.ChanData[6]	4.0		Float	REAL
Of8h0Input.ChanData[7]	4.0		Float	REAL
Of8h0Input.CST	{...}	{...}	Hex	DINT[2]
Of8h0Input.TimeStamp	26618		Decimal	INT
Of8h0Input.HartData	{...}	{...}	Hex	SINT[40]

Output Tags Overview (OF8H Only)

The output tags are used to control the output voltage or current level, depending on the configuration settings, of each individual channel.

Note: The local output tags (i.e. Local:e:O) are automatically created when the module was copied from the sample project in step 1.

Note: The defined output tags (Of8h0Output) are copied to the local output tags (Local:e:O) by the ladder in step 4.

Of8h0Output	{...}	{...}		Of8hOutputBlock
Of8h0Output.Ch0Data	4.0		Float	REAL
Of8h0Output.Ch1Data	4.0		Float	REAL
Of8h0Output.Ch2Data	4.0		Float	REAL
Of8h0Output.Ch3Data	4.0		Float	REAL
Of8h0Output.Ch4Data	4.0		Float	REAL
Of8h0Output.Ch5Data	4.0		Float	REAL
Of8h0Output.Ch6Data	4.0		Float	REAL
Of8h0Output.Ch7Data	4.0		Float	REAL

Configuration, Data, and Status Tags for the 1756sc-IF8H

Read this chapter to:

- Send configuration data to the module
- Configuration tags
- Module filter selection
- Module input tags
- Accessing the module tags
- Changing configuration information at the tags

This chapter outlines the detailed settings for the 1756sc-IF8H. These settings determine the modules input types, filter frequencies, scan rates, and various attributes. Detailed descriptions of these settings are available in the Tag Definition section of this chapter.

Note: The following format is used to describe tags

Tag Name	Data Type	Style	Description

Send Configuration Data to the Module

After changing the configuration tags in this chapter you must then send them to the module. To do this you may perform any of these operations:

1. Inhibit then un-inhibit the module via the module properties dialog, Connection Tab

2. Reset the module via the modules properties dialog. Module Info tab.

3. Reset the module via ladder logic. See the “DoReset” rung in the sample ladder located in Chapter 8.

4. Perform a “Set Attribute All” or Module Reconfigure message instruction via ladder logic. Refer to Chapter 8 for information about the “DoSetAttrAll” command.

Note: If an invalid configuration is sent to the module a connection error will occur. See chapter 9 for a list of error codes.

Configuration Tags for the 1756sc-IF8H

The following tags allow for custom configuration of the module. These tags can be found within the controller scope under the tag name If8h0Configuration.

Table 5.1a

Tag Name	Data Type	Style	Description
If8h0Configuration	If8hConfigurationBlock	NA	Module configuration
If8h0Configuration.ConfigRevision	SINT	DEC	Controls multiple owner connections. 0 = Always connect, overwrite w/new configuration. 1 = Only connect if configuration matches existing configuration.
If8h0Configuration.AdcFilter	SINT	DEC	0 = 10Hz 4 = 250Hz 1 = 50Hz 5 = 1000Hz 2 = 60Hz 6 = 20Hz 3 = 100Hz 7 = 15Hz
If8h0Configuration.RTS	INT	DEC	The time in milliseconds that updated input data is to be sent from the module to the controller. 18 to 10000 msec
If8h0Configuration.ChXConfigBits[1]	SINT	BIN	Channel configuration settings
If8h0Configuration.ChXConfigBits.0[1]	BOOL	DEC	Unused Set to zero
If8h0Configuration.ChXConfigBits.1[1]	BOOL	DEC	Unused Set to zero
If8h0Configuration.ChXConfigBits.2[1]	BOOL	DEC	Unused Set to zero
If8h0Configuration.ChXConfigBits.3[1]	BOOL	DEC	Unused Set to zero
If8h0Configuration.ChXConfigBits.4[1]	BOOL	DEC	Enables latching for the rate alarm. Latching causes the rate alarm to remain set until an unlatch service is explicitly sent to the channel or alarm. 0 = Rate alarm unlatched 1 = Rate alarm latched
If8h0Configuration.ChXConfigBits.5[1]	BOOL	DEC	Enables latching for all four process alarms: low, low low, high, high high. Latching causes the process alarm to remain set until an unlatch service is explicitly sent to the channel or alarm. 0 = Alarms unlatched 1 = Alarms latched
If8h0Configuration.ChXConfigBits.6[1]	BOOL	DEC	Disable alarms 0 = Enabled 1 = Disabled
If8h0Configuration.ChXConfigBits.7[1]	BOOL	DEC	Enable HART Communication 0 = Enabled 1 = Disabled

[1] X represents the module channel number (0 to 7).

Table 5.1b

Tag Name	Data Type	Style	Description
If8h0Configuration.ChXInputRange[1]	SINT	DEC	Configures the channel's input range. Input ranges include: 0 = -10.25 to 10.25V 1 = 0 to 5.125V 2 = 0 to 10.25V 3 = 0 to 20.58 mA 4 = 3.42 mA to 20.58 mA
If8h0Configuration.ChXDigitalFilter[1]	INT	DEC	A non-zero value enables the filter, providing a time constant in milliseconds used in a first order lag filter to smooth the input signal. If non-zero, must be greater than twice the RTS rate. 0 to 20100 msec
If8h0Configuration.ChXRateAlarmLimit[1]	REAL	FLOAT	The trigger point for the rate alarm status bit which will set if the input signal changes at a rate faster than the configured rate alarm. Configured in user scaling units per second.
If8h0Configuration.ChXLowSignal[1]	REAL	FLOAT	One of four points used in scaling. The low signal represents the minimum input range.
If8h0Configuration.ChXHighSignal[1]	REAL	FLOAT	One of four points used in scaling. The high signal represents the maximum input range.
If8h0Configuration.ChXLowEngineering[1]	REAL	FLOAT	One of four points used in scaling. The low engineering represents the minimum scaled range.
If8h0Configuration.ChXHighEngineering[1]	REAL	FLOAT	One of four points used in scaling. The high engineering represents the maximum scaled range.
If8h0Configuration.ChXLowAlarm[1]	REAL	FLOAT	The low process alarm trigger point. Enter the value in terms of engineering units.
If8h0Configuration.ChXHighAlarm[1]	REAL	FLOAT	The high process alarm trigger point. Enter the value in terms of engineering units.
If8h0Configuration.ChXLowLowAlarm[1]	REAL	FLOAT	The low low process alarm trigger point. Enter the value in terms of engineering units.

[1] X represents the module channel number (0 to 7).

Table 5.1c

Tag Name	Data Type	Style	Description
If8h0Configuration.ChXHighHighAlarm[1]	REAL	FLOAT	The high high process alarm trigger point. Enter the value in terms of engineering units.
If8h0Configuration.ChXAlarmDeadband[1]	REAL	FLOAT	Forms a deadband around the process alarms which causes the corresponding process alarm status bit to remain set until the input moves beyond the trigger point by greater than the amount of the alarm deadband.
If8h0Configuration.ChXCalBias[1]	REAL	FLOAT	Offset to add to channel analog value
If8h0Configuration.HandleTimeout	INT	DEC	There is a handle timeout associated with the final reply message. After the module obtaines the requested information from the HART device, it will start the HandleTimeout timer (duration defined by the HandleTimeout tag). The replay message will be kept in memory during the HandleTimeout 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. HandleTimeout is in the range of 1 to 255 seconds.
If8h0Configuration.ModuleConfigBits	INT	BIN	Module function settings
If8h0Configuration.ModuleConfigBits.0	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.1	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.2	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.3	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.4	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.5	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.6	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.7	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.8	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.9	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.10	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.11	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.12	BOOL	DEC	Reserved set to zero
If8h0Configuration.ModuleConfigBits.13	BOOL	DEC	Reserved set to zero

[1] X represents the module channel number (0 to 7).

Table 5.1d

Tag Name	Data Type	Style	Description
If8h0Configuration.ModuleConfigBits.14[2]	BOOL	DEC	Bits 14 and 15 determine how often the pass-through command is serviced by the module. 15,14 = 00 = pass-through serviced once per two channels scanned 15,14 = 01 = pass-through serviced once per module scan 15,14 = 1X = pass-through serviced once per channel scan
If8h0Configuration.ModuleConfigBits.15[2]	BOOL	DEC	Bits 14 and 15 determine how often the pass-through command is serviced by the module. 15,14 = 00 = pass-through serviced once per two channels scanned 15,14 = 01 = pass-through serviced once per module scan 15,14 = 1X = pass-through serviced once per channel scan

[2] The configuration of bits 14 and 15 directly effect the update time of the HART data acquisition for the module, for example if bit 15 is set, the HART update time for each channel will be doubled.

Module Filter Selection

Module Update Time

The module update time is defined as the time required for the module to sample and convert the input signals of all enabled input channels and make the resulting data values available to the processor. The update time is influenced by the input type and filter frequency configuration settings. The following table shows associated time adders based on frequency selection.

Filter	Update Time
10Hz	61 ms per channel
15Hz	41 ms per channel
20Hz	31 ms per channel
50Hz	11 ms per channel
60Hz	11 ms per channel
100Hz	7 ms per channel
250Hz	3.5 ms per channel
1000Hz	2.25 ms per channel

ADC Filter:

The module uses a ADC filter that provides high frequency noise rejection for the input signals. The ADC filter is programmable, allowing you to select from eight filter frequencies for each channel.

Selecting a low value (i.e. 10 Hz) for the filter frequency provides the best noise rejection for a channel, but it also increases the channel update time. Selecting a high value for the filter frequency provides lower noise rejection, but decreases the channel update time.

The module filter is a built-in feature of the Analog-to-Digital convertor which attenuates the input signal beginning at the specified frequency.

In addition to frequency rejection, a by-product of the filter selection is the minimum sample rate (RTS) that is available. For example, the 1000Hz selection will not attenuate any frequencies less than 1000Hz and will allow sampling of all 8 channels within 18ms. But the 10Hz selection will reject all frequencies above 10Hz and will only allow sampling all 8 channels within 488ms. Table 5.2 lists the minimum RTS rate associated with each ADC filter setting. Refer to table 5.1a to configure this feature.

Table 5.2

ADC Filter (Hz)	Minimum RTS (ms)
10	488
15	328
20	248
50	88
60	88
100	56
250	28
1000	18

Digital Filter

The digital filter smooths input data noise transients on each input channel. This value specifies the time constant for a digital first order lag filter on the input. It is specified in units of milliseconds. A value of 0 disables the filter.

The digital filter equation is a classic first order lag equation.

$Yn = Yprevn + (dT/(dT + TA) * (Xn – Yprevn))$

Yn = Filtered peak voltage (PV)

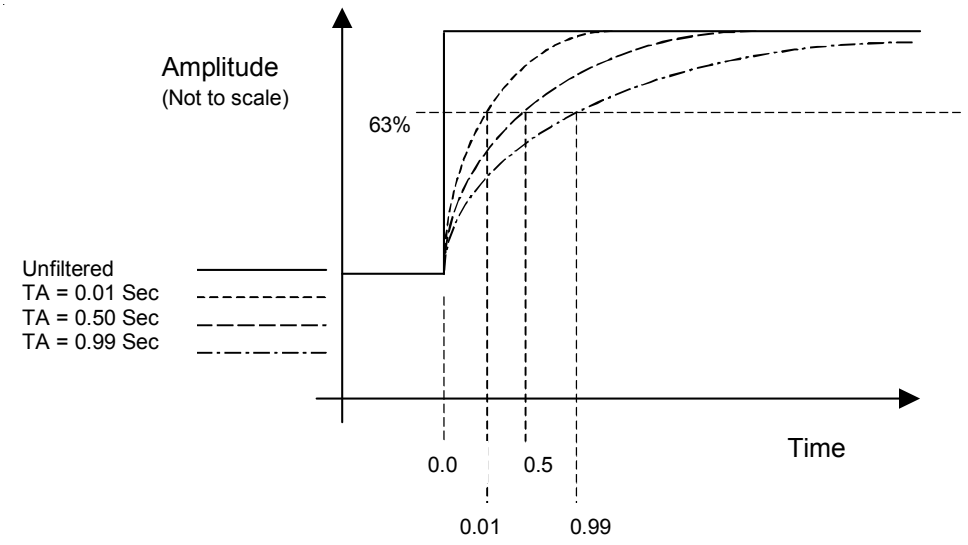
Yprevn = Previous Yn

dT = Time expired since previous Yn (Seconds)

TA = User specified digital filter time constant (Seconds)

Xn = Current value, unfiltered peak voltage (PV)

Using a step input change to illustrate the filter response, as shown below, you can see that when the digital filter time constant elapses, 63.2% of the total response is reached, each additional time constant achieves 63.2% of the remaining response.



Input Tags

The following fault and status reporting and module data sections allow monitoring of faults, status, and input data from the module. These tags can be found within the If8h0Input controller tag.

Table 5.3a

Tag Name	Data Type	Style	Description
If8h0Input	If8h0InputBlock	NA	Module input data
If8h0Input.ModuleStatus	DINT	BIN	General module status
If8h0Input.ModuleStatus.0	BOOL	DEC	Channel 0 general fault status 0 = No fault 1 = Fault
If8h0Input.ModuleStatus.1	BOOL	DEC	Channel 1 general fault status 0 = No fault 1 = Fault
If8h0Input.ModuleStatus.2	BOOL	DEC	Channel 2 general fault status 0 = No fault 1 = Fault
If8h0Input.ModuleStatus.3	BOOL	DEC	Channel 3 general fault status 0 = No fault 1 = Fault
If8h0Input.ModuleStatus.4	BOOL	DEC	Channel 4 general fault status 0 = No fault 1 = Fault
If8h0Input.ModuleStatus.5	BOOL	DEC	Channel 5 general fault status 0 = No fault 1 = Fault
If8h0Input.ModuleStatus.6	BOOL	DEC	Channel 6 general fault status 0 = No fault 1 = Fault
If8h0Input.ModuleStatus.7	BOOL	DEC	Channel 7 general fault status 0 = No fault 1 = Fault
If8h0Input.ModuleStatus.8	BOOL	DEC	Channel 0 broken wire 0 = Connection 1 = Open circuit
If8h0Input.ModuleStatus.9	BOOL	DEC	Channel 1 broken wire 0 = Connection 1 = Open circuit
If8h0Input.ModuleStatus.10	BOOL	DEC	Channel 2 broken wire 0 = Connection 1 = Open circuit
If8h0Input.ModuleStatus.11	BOOL	DEC	Channel 3 broken wire 0 = Connection 1 = Open circuit
If8h0Input.ModuleStatus.12	BOOL	DEC	Channel 4 broken wire 0 = Connection 1 = Open circuit

Table 5.3b

Tag Name	Data Type	Style	Description
If8h0Input.ModuleStatus.13	BOOL	DEC	Channel 5 broken wire 0 = Connection 1 = Open circuit
If8h0Input.ModuleStatus.14	BOOL	DEC	Channel 6 broken wire 0 = Connection 1 = Open circuit
If8h0Input.ModuleStatus.15	BOOL	DEC	Channel 7 broken wire 0 = Connection 1 = Open circuit
If8h0Input.ModuleStatus.16	BOOL	DEC	Channel 0 HART out-of-service 0 = In service 1 = Out-of-service
If8h0Input.ModuleStatus.17	BOOL	DEC	Channel 1 HART out-of-service 0 = In service 1 = Out-of-service
If8h0Input.ModuleStatus.18	BOOL	DEC	Channel 2 HART out-of-service 0 = In service 1 = Out-of-service
If8h0Input.ModuleStatus.19	BOOL	DEC	Channel 3 HART out-of-service 0 = In service 1 = Out-of-service
If8h0Input.ModuleStatus.20	BOOL	DEC	Channel 4 HART out-of-service 0 = In service 1 = Out-of-service
If8h0Input.ModuleStatus.21	BOOL	DEC	Channel 5 HART out-of-service 0 = In service 1 = Out-of-service
If8h0Input.ModuleStatus.22	BOOL	DEC	Channel 6 HART out-of-service 0 = In service 1 = Out-of-service
If8h0Input.ModuleStatus.23	BOOL	DEC	Channel 7 HART out-of-service 0 = In service 1 = Out-of-service
If8h0Input.ModuleStatus.24	BOOL	DEC	Unused
If8h0Input.ModuleStatus.25	BOOL	DEC	Calibration fault 0 = No fault 1 = Fault
If8h0Input.ModuleStatus.26	BOOL	DEC	Calibration in progress 0 = Calibration NOT active 1 = Calibrating
If8h0Input.ModuleStatus.27	BOOL	DEC	Unused
If8h0Input.ModuleStatus.28	BOOL	DEC	Unused
If8h0Input.ModuleStatus.29	BOOL	DEC	Unused
If8h0Input.ModuleStatus.30	BOOL	DEC	Unused
If8h0Input.ModuleStatus.31	BOOL	DEC	Analog group fault 0 = No fault 1 = Fault

Table 5.3c

Tag Name	Data Type	Style	Description
If8h0Input.ChanStatus	SINT[8]	BIN	An array of 8 SINT which display general status for each channel.
If8h0Input.ChanStatus[X][1]	SINT	BIN	General channel status byte
If8h0Input.ChanStatus[X].0[1]	BOOL	DEC	User value exceeded high high limit. 0 = No alarm 1 = Alarm
If8h0Input.ChanStatus[X].1[1]	BOOL	DEC	User value exceeded low low limit. 0 = No alarm 1 = Alarm
If8h0Input.ChanStatus[X].2[1]	BOOL	DEC	User value exceeded high limit. 0 = No alarm 1 = Alarm
If8h0Input.ChanStatus[X].3[1]	BOOL	DEC	User value exceeded low limit. 0 = No alarm 1 = Alarm
If8h0Input.ChanStatus[X].4[1]	BOOL	DEC	Specified ramp rate exceeded. 0 = No alarm 1 = Alarm
If8h0Input.ChanStatus[X].5[1]	BOOL	DEC	Over Range alarm 0 = No alarm 1 = Alarm
If8h0Input.ChanStatus[X].6[1]	BOOL	DEC	Under Range alarm 0 = No alarm 1 = Alarm
If8h0Input.ChanStatus[X].7[1]	BOOL	DEC	Channel calibration fault 0 = No fault 1 = Fault
If8h0Input.ChanData	REAL[8]	FLOAT	An array of 8 floating point registers which display the analog data for each channel in engineering units.
If8h0Input.ChanData[X][1]	REAL	FLOAT	Channel analog data in engineering units

[1] X represents the module channel number (0 to 7).

Table 5.3d

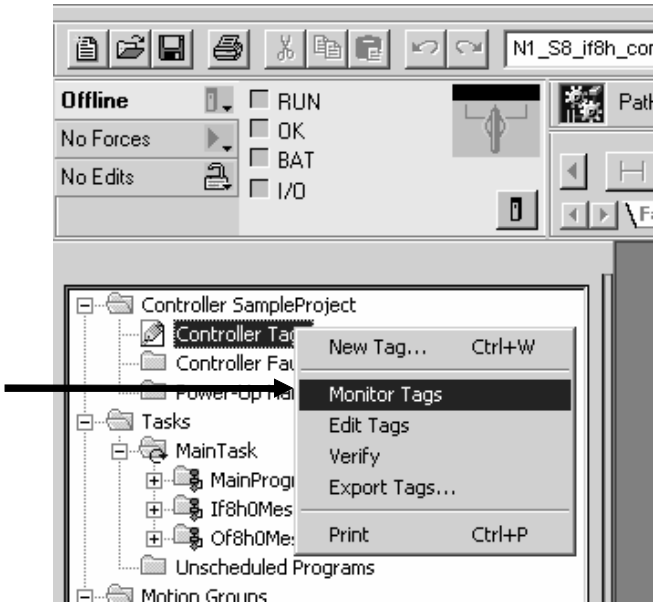
Tag Name	Data Type	Style	Description
If8h0Input.CST	DINT[2]	DEC	This is the timestamp taken at the time the input data was sampled, which is in terms of coordinated system time. This is a 64 bit quantity in microseconds coordinated across the rack. This must be addressed in 32 bit segments as an array.
If8h0Input.TimeStamp	INT	DEC	This is the timestamp taken at time the input data was sampled, which is shown in milliseconds relative solely to the individual module.
If8h0Input.HartData	SINT[40]	DEC	An array of 40 bytes which contains HART packet data for the entir more details.

Accessing The Module Tags

When you access tags to change configuration or monitor the I/O data exchange, you have two options.:

- **Monitor tags** - option allows you to view tags and change their values.
- **Edit tags** - option allows you to add or delete tags, but not to change their values.

1.) Select Controller Tags
2.) Right-click to display the menu
3.) Select Monitor Tags



You can view tags here.

Click on the tag name of the data structure you want to see

Tag Name	Value	Force Mask	Style	Type	Description
If8h0ChannelNumber	0		Decimal	DINT	
If8h0Configuration	{...}	{...}		If8hConfigurationBlock	
If8h0Input	{...}	{...}		If8hInputBlock	
If8h0Packet0	{...}	{...}		PACKET0[8,1]	
If8h0Packet1	{...}	{...}		PACKET1[8,1]	
If8h0Packet2	{...}	{...}		PACKET2[8,1]	
If8h0Packet3	{...}	{...}		PACKET3[8,1]	
If8h0Packet4	{...}	{...}		PACKET4[8,1]	
If8h0PacketNumber	0		Decimal	DINT	
If8h0PassThruQryMsg	{...}	{...}		MESSAGE	
If8h0PassThruQryRX	{...}	{...}	Decimal	SINT[260]	
If8h0PassThruQryTX	{...}	{...}	Decimal	SINT[6]	
If8h0PassThruReqMsg	{...}	{...}		MESSAGE	
If8h0PassThruReqRX	{...}	{...}	Decimal	SINT[6]	
If8h0PassThruReqTX	{...}	{...}	Decimal	SINT[260]	
Local:1:C	{...}	{...}		AB:1756_MODULE:C:0	
Local:1:I	{...}	{...}		AB:1756_MODULE_DINT_96Bytes:I:0	
Local:2:C	{...}	{...}		AB:1756_MODULE:C:0	
Local:2:I	{...}	{...}		AB:1756_MODULE_DINT_96Bytes:I:0	
Local:2:O	{...}	{...}		AB:1756_MODULE_DINT_32Bytes:O:0	
Of8h0ChannelNumber	7		Decimal	DINT	
Of8h0Configuration	{...}	{...}		Of8hConfigurationBlock	
Of8h0Input	{...}	{...}		Of8hInputBlock	
Of8h0Output	{...}	{...}		Of8hOutputBlock	
Of8h0Packet0	{...}	{...}		PACKET0[8,1]	
Of8h0Packet1	{...}	{...}		PACKET1[8,1]	
Of8h0Packet2	{...}	{...}		PACKET2[8,1]	
Of8h0Packet3	{...}	{...}		PACKET3[8,1]	
Of8h0Packet4	{...}	{...}		PACKET4[8,1]	
Of8h0PacketNumber	0		Decimal	DINT	
Of8h0PassThruQryMsg	{...}	{...}		MESSAGE	
Of8h0PassThruQryRX	{...}	{...}	Decimal	SINT[260]	
Of8h0PassThruQryTX	{...}	{...}	Decimal	SINT[6]	
Of8h0PassThruReqMsg	{...}	{...}		MESSAGE	
Of8h0PassThruReqRX	{...}	{...}	Decimal	SINT[6]	
Of8h0PassThruReqTX	{...}	{...}	Decimal	SINT[260]	

Configuration information is listed for each channel on the module

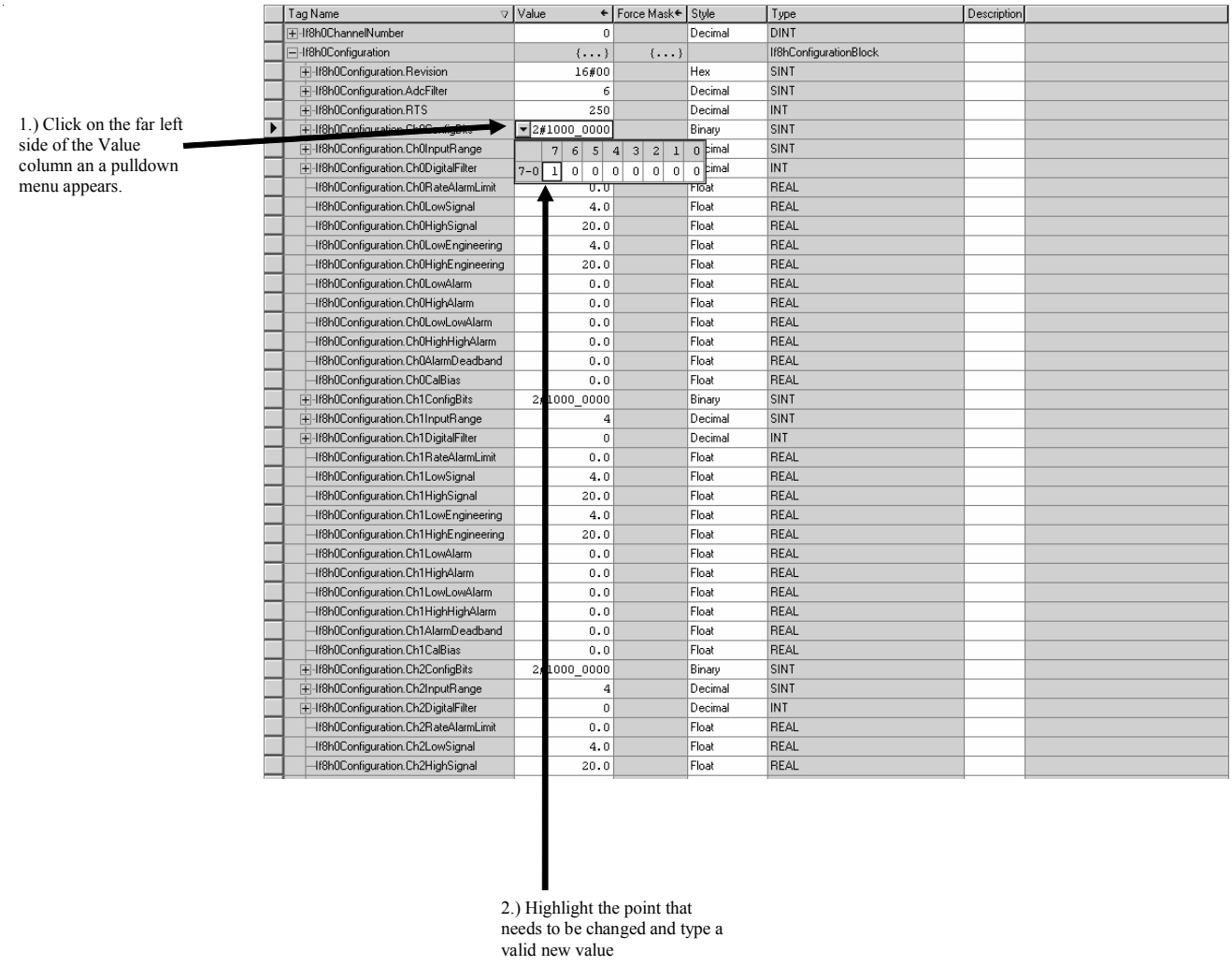
Tag Name	Value	Force Mask	Style	Type
If8h0ChannelNumber	0		Decimal	DINT
If8h0Configuration	{...}	{...}		If8hConfigurationBlock
If8h0Configuration.Revision	16#00		Hex	SINT
If8h0Configuration.AdcFilter	6		Decimal	SINT
If8h0Configuration.RTS	250		Decimal	INT
If8h0Configuration.Ch0ConfigBits	2#1000_0000		Binary	SINT
If8h0Configuration.Ch0InputRange	4		Decimal	SINT
If8h0Configuration.Ch0DigitalFilter	0		Decimal	INT
If8h0Configuration.Ch0RateAlarmLimit	0.0		Float	REAL
If8h0Configuration.Ch0LowSignal	4.0		Float	REAL
If8h0Configuration.Ch0HighSignal	20.0		Float	REAL
If8h0Configuration.Ch0LowEngineering	4.0		Float	REAL
If8h0Configuration.Ch0HighEngineering	20.0		Float	REAL
If8h0Configuration.Ch0LowAlarm	0.0		Float	REAL
If8h0Configuration.Ch0HighAlarm	0.0		Float	REAL
If8h0Configuration.Ch0LowLowAlarm	0.0		Float	REAL
If8h0Configuration.Ch0HighHighAlarm	0.0		Float	REAL
If8h0Configuration.Ch0AlarmDeadband	0.0		Float	REAL
If8h0Configuration.Ch0CalBias	0.0		Float	REAL
If8h0Configuration.Ch1ConfigBits	2#1000_0000		Binary	SINT
If8h0Configuration.Ch1InputRange	4		Decimal	SINT
If8h0Configuration.Ch1DigitalFilter	0		Decimal	INT
If8h0Configuration.Ch1RateAlarmLimit	0.0		Float	REAL
If8h0Configuration.Ch1LowSignal	4.0		Float	REAL
If8h0Configuration.Ch1HighSignal	20.0		Float	REAL
If8h0Configuration.Ch1LowEngineering	4.0		Float	REAL

Changing Configuration Information at the Tags

There are two ways to change the configuration:

- Use a pulldown menu
- Highlight the value of a particular feature for a particular point and type a new value

Pulldown menu

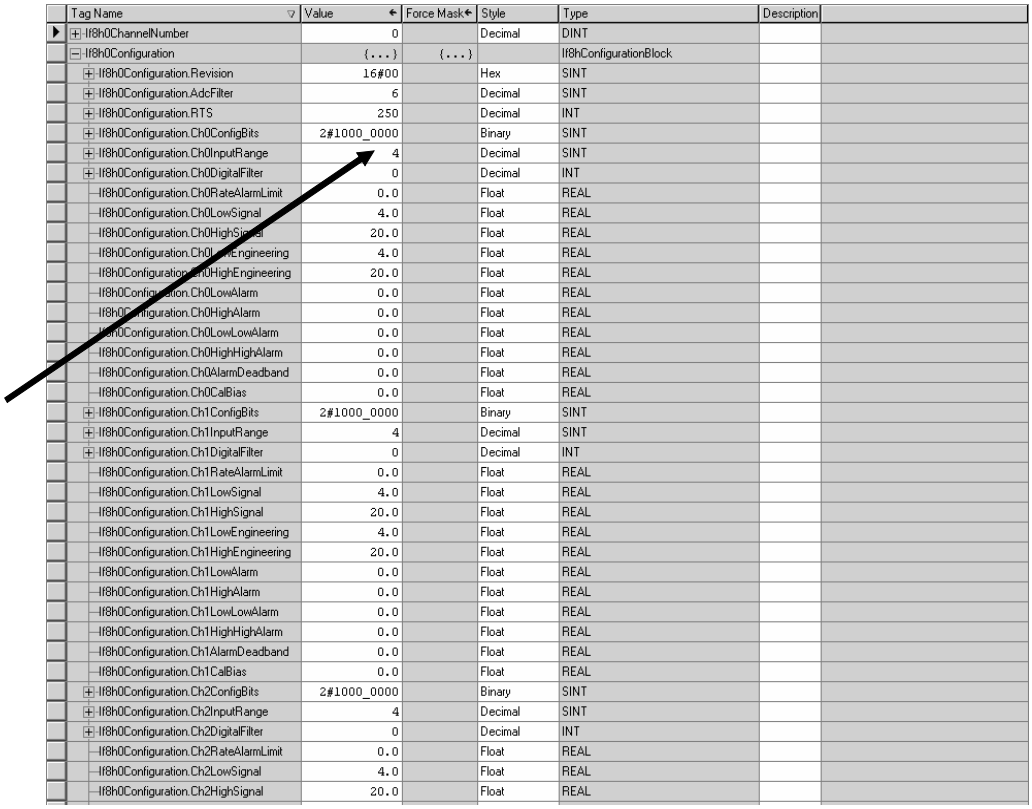


Tag Name	Value	Force Mask	Style	Type	Description
If8h0ChannelNumber	0		Decimal	DINT	
If8h0Configuration	{...}	{...}		If8hConfigurationBlock	
If8h0Configuration.Revision	16#00		Hex	SINT	
If8h0Configuration.AdcFilter	6		Decimal	SINT	
If8h0Configuration.RTS	250		Decimal	INT	
If8h0Configuration.Ch0ConfigBits	2#1000_0000		Binary	SINT	
If8h0Configuration.Ch0InputRange			cimal	SINT	
If8h0Configuration.Ch0DigitalFilter			cimal	INT	
If8h0Configuration.Ch0RateAlarmLimit	0.0		Float	REAL	
If8h0Configuration.Ch0LowSignal	4.0		Float	REAL	
If8h0Configuration.Ch0HighSignal	20.0		Float	REAL	
If8h0Configuration.Ch0LowEngineering	4.0		Float	REAL	
If8h0Configuration.Ch0HighEngineering	20.0		Float	REAL	
If8h0Configuration.Ch0LowAlarm	0.0		Float	REAL	
If8h0Configuration.Ch0HighAlarm	0.0		Float	REAL	
If8h0Configuration.Ch0LowLowAlarm	0.0		Float	REAL	
If8h0Configuration.Ch0HighHighAlarm	0.0		Float	REAL	
If8h0Configuration.Ch0AlarmDeadband	0.0		Float	REAL	
If8h0Configuration.Ch0CalBias	0.0		Float	REAL	
If8h0Configuration.Ch1ConfigBits	2#1000_0000		Binary	SINT	
If8h0Configuration.Ch1InputRange	4		Decimal	SINT	
If8h0Configuration.Ch1DigitalFilter	0		Decimal	INT	
If8h0Configuration.Ch1RateAlarmLimit	0.0		Float	REAL	
If8h0Configuration.Ch1LowSignal	4.0		Float	REAL	
If8h0Configuration.Ch1HighSignal	20.0		Float	REAL	
If8h0Configuration.Ch1LowEngineering	4.0		Float	REAL	
If8h0Configuration.Ch1HighEngineering	20.0		Float	REAL	
If8h0Configuration.Ch1LowAlarm	0.0		Float	REAL	
If8h0Configuration.Ch1HighAlarm	0.0		Float	REAL	
If8h0Configuration.Ch1LowLowAlarm	0.0		Float	REAL	
If8h0Configuration.Ch1HighHighAlarm	0.0		Float	REAL	
If8h0Configuration.Ch1AlarmDeadband	0.0		Float	REAL	
If8h0Configuration.Ch1CalBias	0.0		Float	REAL	
If8h0Configuration.Ch2ConfigBits	2#1000_0000		Binary	SINT	
If8h0Configuration.Ch2InputRange	4		Decimal	SINT	
If8h0Configuration.Ch2DigitalFilter	0		Decimal	INT	
If8h0Configuration.Ch2RateAlarmLimit	0.0		Float	REAL	
If8h0Configuration.Ch2LowSignal	4.0		Float	REAL	
If8h0Configuration.Ch2HighSignal	20.0		Float	REAL	

	7	6	5	4	3	2	1	0
7-0	1	0	0	0	0	0	0	0

Highlight value

1.) Highlight the value of the feature you want to change
2.) **Type in the valid new value.**



Tag Name	Value	Force Mask	Style	Type	Description
If8h0ChannelNumber	0		Decimal	DINT	
If8h0Configuration	{...}	{...}		If8hConfigurationBlock	
If8h0Configuration.Revision	16#00		Hex	SINT	
If8h0Configuration.AdcFilter	6		Decimal	SINT	
If8h0Configuration.RTS	250		Decimal	INT	
If8h0Configuration.Ch0ConfigBits	2#1000_0000		Binary	SINT	
If8h0Configuration.Ch0InputRange	4		Decimal	SINT	
If8h0Configuration.Ch0DigitalFilter	0		Decimal	INT	
If8h0Configuration.Ch0RateAlarmLimit	0.0		Float	REAL	
If8h0Configuration.Ch0LowSignal	4.0		Float	REAL	
If8h0Configuration.Ch0HighSignal	20.0		Float	REAL	
If8h0Configuration.Ch0LowEngineering	4.0		Float	REAL	
If8h0Configuration.Ch0HighEngineering	20.0		Float	REAL	
If8h0Configuration.Ch0LowAlarm	0.0		Float	REAL	
If8h0Configuration.Ch0HighAlarm	0.0		Float	REAL	
If8h0Configuration.Ch0LowLowAlarm	0.0		Float	REAL	
If8h0Configuration.Ch0HighHighAlarm	0.0		Float	REAL	
If8h0Configuration.Ch0AlarmDeadband	0.0		Float	REAL	
If8h0Configuration.Ch0CalBias	0.0		Float	REAL	
If8h0Configuration.Ch1ConfigBits	2#1000_0000		Binary	SINT	
If8h0Configuration.Ch1InputRange	4		Decimal	SINT	
If8h0Configuration.Ch1DigitalFilter	0		Decimal	INT	
If8h0Configuration.Ch1RateAlarmLimit	0.0		Float	REAL	
If8h0Configuration.Ch1LowSignal	4.0		Float	REAL	
If8h0Configuration.Ch1HighSignal	20.0		Float	REAL	
If8h0Configuration.Ch1LowEngineering	4.0		Float	REAL	
If8h0Configuration.Ch1HighEngineering	20.0		Float	REAL	
If8h0Configuration.Ch1LowAlarm	0.0		Float	REAL	
If8h0Configuration.Ch1HighAlarm	0.0		Float	REAL	
If8h0Configuration.Ch1LowLowAlarm	0.0		Float	REAL	
If8h0Configuration.Ch1HighHighAlarm	0.0		Float	REAL	
If8h0Configuration.Ch1AlarmDeadband	0.0		Float	REAL	
If8h0Configuration.Ch1CalBias	0.0		Float	REAL	
If8h0Configuration.Ch2ConfigBits	2#1000_0000		Binary	SINT	
If8h0Configuration.Ch2InputRange	4		Decimal	SINT	
If8h0Configuration.Ch2DigitalFilter	0		Decimal	INT	
If8h0Configuration.Ch2RateAlarmLimit	0.0		Float	REAL	
If8h0Configuration.Ch2LowSignal	4.0		Float	REAL	
If8h0Configuration.Ch2HighSignal	20.0		Float	REAL	

Configuration, Data, and Status Tags for the 1756sc-OF8H

Read this chapter to:

- Send configuration data to the module
- Configuration tags
- Input tags
- Output tags
- Accessing the module tags
- Changing configuration information at the tags

This chapter outlines the detailed settings for the 1756sc-OF8H. These settings determine the modules input types, scan rates, and various attributes. Detailed descriptions of these settings are available in the Tag Definition section of this chapter.

Note: The following format is used to describe tags

Tag Name	Data Type	Style	Description

Send Configuration Data to the Module

After changing the configuration tags in this chapter you must then send them to the module. To do this you may perform any of these operations:

1. Inhibit then un-inhibit the module via the module properties dialog, Connection Tab

2. Reset the module via the modules properties dialog. Module Info tab.

3. Reset the module via ladder logic. See the “DoReset” rung in the sample ladder located in Chapter 8.

4. Perform a “Set Attribute All” or Module Reconfigure message instruction via ladder logic. Refer to Chapter 8 for information about the “DoSetAttrAll” command.

Note: If an invalid configuration is sent to the module a connection error will occur. See chapter 9 for a list of error codes.

Configuration Tags for the 1756sc-OF8H

The following tags allow for custom configuration of the module. These tags can be found within the controller scope under the tag name Of8h0Configuration.

Table 6.1a

Tag Name	Data Type	Style	Description
Of8h0Configuration	Of8hConfigurationBlock	NA	Module configuration
Of8h0Configuration.ConfigRevision	SINT	DEC	Controls multiple owner connections. 0 = Always connect, overwrite w/new configuration. 1 = Only connect if configuration matches existing configuration.
Of8h0Configuration.ProgToFaultEn	SINT	DEC	The program to fault enable bit determines how the outputs should behave if a communications fault were to occur while the output module is in the program mode. 0 = Outputs will remain in their configured program state despite a communications fault occurring. 1 = will cause the outputs to transition to their programmed fault state if a communications fault occurs while in the program state.
Of8h0Configuration.Spare	INT	DEC	Unused Set to zero
Of8h0Configuration.ChXConfigBits[1]	INT	BIN	This is a set of individual channel configuration bits which enable various functions for the associated channel.
Of8h0Configuration.ChXConfigBits.0[1]	BOOL	DEC	Enables ramping of the output value to a user defined fault value (ChXFaultValue) 0 = Feature disabled 1 = During a fault condition the associated channel will ramp to the value stored in the ChXFaultValue tag at a rate defined by the value stored in the ChXMaxRampRate.
Of8h0Configuration.ChXConfigBits.1[1]	BOOL	DEC	Enables ramping of the output value to a user defined idle value (ChXIdleValue) 0 = Feature disabled 1 = When the PLC is in program mode the associated channel will ramp to the value stored in the ChXIdleValue tag at a rate defined by the value stored in the ChXMaxRampRate.

[1] X represents the module channel number (0 to 7).

Table 6.1b

Tag Name	Data Type	Style	Description
Of8h0Configuration.ChXConfigBits.2[1]	BOOL	DEC	Enables ramping of the output value when the PLC is in RUN mode. 0 = Feature disabled 1 = When the PLC is in RUN mode the associated channel will ramp to a new output level at a rate defined by the value stored in the ChXMaxRampRate tag.
Of8h0Configuration.ChXConfigBits.3[1]	BOOL	DEC	Selects the behavior of the output channel when transitioning into Program mode. 0 = Hold last state 1 = go to the value stored in the ChXIdleValue tag.
Of8h0Configuration.ChXConfigBits.4[1]	BOOL	DEC	Selects the behavior of the output channel when a communication fault occurs. 0 = Hold last state 1 = go to the value stored in the ChXFaultValue tag.
Of8h0Configuration.ChXConfigBits.5[1]	BOOL	DEC	Enables latching for the clamp limit alarms. 0 = Feature disabled 1 = Clamp limit alarm will remain set until an unlatch service is explicitly sent to the channel or alarm.
Of8h0Configuration.ChXConfigBits.6[1]	BOOL	DEC	Enables latching for all four process alarms: low, low low, high, and high high. 0 = Alarms unlatched 1 = process alarms will remain set until an unlatch service is explicitly sent to the channel or alarm.
Of8h0Configuration.ChXConfigBits.7[1]	BOOL	DEC	Disable alarms 0 = Alarms enabled 1 = Disable all alarms

[1] X represents the module channel number (0 to 7).

Table 6.1c

Tag Name	Data Type	Style	Description
Of8h0Configuration.ChXConfigBits.8[1]	BOOL	DEC	Hold for initialization 0 = Feature disabled 1 = The channel will hold, or not change, until initialized with a value within 0.1% of full scale of its current value when one of the following conditions occurs: Module initial connection (power-up), module transitions from program mode back to run mode, module reestablishes communication after a fault.
Of8h0Configuration.ChXConfigBits.9[1]	BOOL	DEC	Enables HART communication for the channel. 0 = HART disabled 1 = HART Enabled
Of8h0Configuration.ChXConfigBits.10[1]	BOOL	DEC	Unused Set to zero
Of8h0Configuration.ChXConfigBits.11[1]	BOOL	DEC	Unused Set to zero
Of8h0Configuration.ChXConfigBits.12[1]	BOOL	DEC	Unused Set to zero
Of8h0Configuration.ChXConfigBits.13[1]	BOOL	DEC	Unused Set to zero
Of8h0Configuration.ChXConfigBits.14[1]	BOOL	DEC	Unused Set to zero
Of8h0Configuration.ChXConfigBits.15[1]	BOOL	DEC	Unused Set to zero
Of8h0Configuration.ChXOutputRange[1]	INT	DEC	Configures the channel's output range. 0 = +/-10 VDC 1 = 0 to 20mA 2 = 4 to 20mA
Of8h0Configuration.ChXMaxRampRate[1]	FLOAT	REAL	Configures the maximum rate at which the output value may change. Active only if bits 0, 1 or 2 are set in the ChXConfigBits tag. Entered in user scaling units per second.
Of8h0Configuration.ChXFaultValue[1]			Defines the value, in engineering units, the output should take if a communications fault occurs when bit 0 is set in the ChXConfigBits tag.

[1] X represents the module channel number (0 to 7).

Table 6.1d

Tag Name	Data Type	Style	Description
Of8h0Configuration.ChXIdleValue[1]	FLOAT	REAL	Defines the value, in engineering units, the output should take when the connection transitions to Program mode. This function is only active when bit 1 is set in the ChXConfigBits tag.
Of8h0Configuration.ChXLowSignal[1]	FLOAT	REAL	One of four points used in scaling. The low signal represents the minimum output range.
Of8h0Configuration.ChXHighSignal[1]	FLOAT	REAL	One of four points used in scaling. The high signal represents the maximum output range.
Of8h0Configuration.ChXLowEngineering[1]	FLOAT	REAL	One of four points used in scaling. The low engineering represents the minimum scaled range.
Of8h0Configuration.ChXHighEngineering[1]	FLOAT	REAL	One of four points used in scaling. The high engineering represents the maximum scaled range.
Of8h0Configuration.ChXLowClamp[1]	FLOAT	REAL	Defines the minimum analog output value the channel is allowed to achieve.
Of8h0Configuration.ChXHighClamp[1]	FLOAT	REAL	Defines the maximum analog output value the channel is allowed to achieve.
Of8h0Configuration.ChXCalBias[1]	FLOAT	REAL	A user configurable offset added directly to the measured analog value for the channel.
Of8h0Configuration.ChXSlot0Code[1]	SINT	DEC	HART slot 0 transmitter variable assignment code. Refer to chapter 7 for more information.
Of8h0Configuration.ChXSlot1Code[1]	SINT	DEC	HART slot 1 transmitter variable assignment code. Refer to chapter 7 for more information.
Of8h0Configuration.ChXSlot2Code[1]	SINT	DEC	HART slot 2 transmitter variable assignment code. Refer to chapter 7 for more information.
Of8h0Configuration.ChXSlot3Code[1]	SINT	DEC	HART slot 3 transmitter variable assignment code. Refer to chapter 7 for more information.

[1] X represents the module channel number (0 to 7).

Table 6.1e

Tag Name	Data Type	Style	Description
Of8h0Configuration.HandleTimeout	INT	DEC	There is a handle timeout associated with the final reply message. After the module obtaines the requested information from the HART device, it will start the HandleTimeout timer (duration defined by the HandleTimeout tag). The replay message will be kept in memory during the HandleTimeout 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. HandleTimeout is in the range of 1 to 255 seconds.
Of8h0Configuration.ModuleConfigBits	INT	BIN	Module function settings
Of8h0Configuration.ModuleConfigBits.0	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.1	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.2	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.3	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.4	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.5	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.6	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.7	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.8	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.9	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.10	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.11	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.12	BOOL	DEC	Reserved set to zero
Of8h0Configuration.ModuleConfigBits.13	BOOL	DEC	Reserved set to zero

Table 6.1f

Tag Name	Data Type	Style	Description
Of8h0Configuration.ModuleConfigBits.14[2]	BOOL	DEC	Bits 14 and 15 determine how often the pass-through command is serviced by the module. 15,14 = 00 = pass-through serviced once per two channels scanned 15,14 = 01 = pass-through serviced once per module scan 15,14 = 1X = pass-through serviced once per channel scan
Of8h0Configuration.ModuleConfigBits.15[2]	BOOL	DEC	Bits 14 and 15 determine how often the pass-through command is serviced by the module. 15,14 = 00 = pass-through serviced once per two channels scanned 15,14 = 01 = pass-through serviced once per module scan 15,14 = 1X = pass-through serviced once per channel scan

[2] The configuration of bits 14 and 15 directly effect the update time of the HART data acquisition for the module, for example if bit 15 is set, the HART update time for each channel will be doubled.

Input Tags

The following fault and status reporting and module data sections allow monitoring of faults, status, and input data from the module. These tags can be found within the Of8h0Input controller tag.

Table 6.2a

Tag Name	Data Type	Style	Description
Of8h0Input	Of8h0InputBlock	NA	Module input data
Of8h0Input.ModuleStatus	DINT	BIN	General module status
Of8h0Input.ModuleStatus.0	BOOL	DEC	Channel 0 general fault status 0 = No fault 1 = Fault
Of8h0Input.ModuleStatus.1	BOOL	DEC	Channel 1 general fault status 0 = No fault 1 = Fault
Of8h0Input.ModuleStatus.2	BOOL	DEC	Channel 2 general fault status 0 = No fault 1 = Fault
Of8h0Input.ModuleStatus.3	BOOL	DEC	Channel 3 general fault status 0 = No fault 1 = Fault
Of8h0Input.ModuleStatus.4	BOOL	DEC	Channel 4 general fault status 0 = No fault 1 = Fault
Of8h0Input.ModuleStatus.5	BOOL	DEC	Channel 5 general fault status 0 = No fault 1 = Fault
Of8h0Input.ModuleStatus.6	BOOL	DEC	Channel 6 general fault status 0 = No fault 1 = Fault
Of8h0Input.ModuleStatus.7	BOOL	DEC	Channel 7 general fault status 0 = No fault 1 = Fault
Of8h0Input.ModuleStatus.8	BOOL	DEC	Loop output fault 0 = No Fault 1 = Fault
Of8h0Input.ModuleStatus.9	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.10	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.11	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.12	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.13	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.14	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.15	BOOL	DEC	Unused

Table 6.2b

Tag Name	Data Type	Style	Description
Of8h0Input.ModuleStatus.16	BOOL	DEC	Channel 0 HART out-of-service 0 = In service 1 = Out-of-service
Of8h0Input.ModuleStatus.17	BOOL	DEC	Channel 1 HART out-of-service 0 = In service 1 = Out-of-service
Of8h0Input.ModuleStatus.18	BOOL	DEC	Channel 2 HART out-of-service 0 = In service 1 = Out-of-service
Of8h0Input.ModuleStatus.19	BOOL	DEC	Channel 3 HART out-of-service 0 = In service 1 = Out-of-service
Of8h0Input.ModuleStatus.20	BOOL	DEC	Channel 4 HART out-of-service 0 = In service 1 = Out-of-service
Of8h0Input.ModuleStatus.21	BOOL	DEC	Channel 5 HART out-of-service 0 = In service 1 = Out-of-service
Of8h0Input.ModuleStatus.22	BOOL	DEC	Channel 6 HART out-of-service 0 = In service 1 = Out-of-service
Of8h0Input.ModuleStatus.23	BOOL	DEC	Channel 7 HART out-of-service 0 = In service 1 = Out-of-service
Of8h0Input.ModuleStatus.24	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.25	BOOL	DEC	Calibration fault 0 = No fault 1 = Fault
Of8h0Input.ModuleStatus.26	BOOL	DEC	Calibration in progress 0 = Calibration NOT active 1 = Calibrating
Of8h0Input.ModuleStatus.27	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.28	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.29	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.30	BOOL	DEC	Unused
Of8h0Input.ModuleStatus.31	BOOL	DEC	Analog group fault 0 = No fault 1 = Fault

Table 6.2c

Tag Name	Data Type	Style	Description
Of8h0Input.ChanStatus	SINT[8]	BIN	An array of 8 SINT which display general status for each channel.
Of8h0Input.ChanStatus[X][1]	SINT	BIN	General channel status byte
Of8h0Input.ChanStatus[X].0[1]	BOOL	DEC	User value exceeded high clamp limit. 0 = No alarm 1 = Alarm
Of8h0Input.ChanStatus[X].1[1]	BOOL	DEC	User value exceeded low clamp limit. 0 = No alarm 1 = Alarm
Of8h0Input.ChanStatus[X].2[1]	BOOL	DEC	Specified ramp rate exceeded. 0 = No alarm 1 = Alarm
Of8h0Input.ChanStatus[X].3[1]	BOOL	DEC	Channel holding last output value. 0 = Normal 1 = Holding
Of8h0Input.ChanStatus[X].4[1]	BOOL	DEC	Channel calibration fault 0 = No fault 1 = Fault
Of8h0Input.ChanStatus[X].5[1,2]	BOOL	DEC	Invalid or NaN channel value 0 = No alarm 1 = Alarm
Of8h0Input.ChanStatus[X].6[1]	BOOL	DEC	Unused
Of8h0Input.ChanStatus[X].7[1]	BOOL	DEC	Open wire fault (Current Only) 0 = No fault 1 = Fault
Of8h0Input.ChanData	REAL[8]	FLOAT	An array of 8 floating point registers which display the analog data for each channel in engineering units.
Of8h0Input.ChanData[X][1]	REAL	FLOAT	Channel analog data in engineering units

[1] X represents the module channel number (0 to 7).

[2] An invalid data value is being used for the associated channel's output tag (i.e. Of8h0.Output.ChXData)

Table 6.2d

Tag Name	Data Type	Style	Description
Of8h0Input.CST	DINT[2]	DEC	This is the timestamp taken at the time the input data was sampled, which is in terms of coordinated system time. This is a 64 bit quantity in microseconds coordinated across the rack. This must be addressed in 32 bit segments as an array.
Of8h0Input.TimeStamp	INT	DEC	This is the timestamp taken at time the input data was sampled, which is shown in milliseconds relative solely to the individual module.
Of8h0Input.HartData	SINT[40]	DEC	An array of 40 bytes which contains HART packet data for the entir more details.

Output Tags

The output tags are used to control the analog signals for each channel. See the table below for more details.

Table 6.3

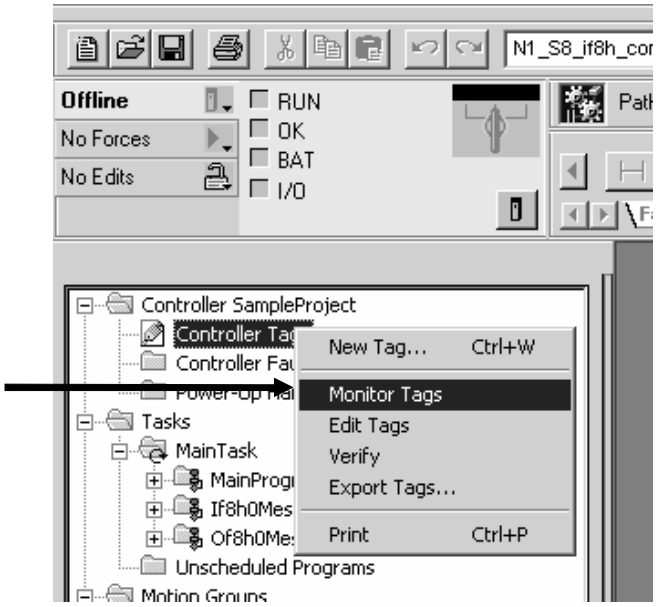
Tag Name	Data Type	Style	Description
Of8h0Output	Of8hOutputBlock	NA	Module output data
Of8h0Output.Ch0Data	FLOAT	REAL	Controls the analog output signal for channel 0
Of8h0Output.Ch1Data	FLOAT	REAL	Controls the analog output signal for channel 1
Of8h0Output.Ch2Data	FLOAT	REAL	Controls the analog output signal for channel 2
Of8h0Output.Ch3Data	FLOAT	REAL	Controls the analog output signal for channel 3
Of8h0Output.Ch4Data	FLOAT	REAL	Controls the analog output signal for channel 4
Of8h0Output.Ch5Data	FLOAT	REAL	Controls the analog output signal for channel 5
Of8h0Output.Ch6Data	FLOAT	REAL	Controls the analog output signal for channel 6
Of8h0Output.Ch7Data	FLOAT	REAL	Controls the analog output signal for channel 7

Accessing The Module Tags

When you access tags to change configuration or monitor the I/O data exchange, you have two options.:

- **Monitor tags** - option allows you to view tags and change their values.
- **Edit tags** - option allows you to add or delete tags, but not to change their values.

1.) Select Controller Tags
2.) Right-click to display the menu
3.) Select Monitor Tags



You can view tags here.

Tag Name	Value	Force Mask	Style	Type	Description
If8h0ChannelNumber	0		Decimal	DINT	
If8h0Configuration	{...}	{...}		If8hConfigurationBlock	
If8h0Input	{...}	{...}		If8hInputBlock	
If8h0Packet0	{...}	{...}		PACKET0[8,1]	
If8h0Packet1	{...}	{...}		PACKET1[8,1]	
If8h0Packet2	{...}	{...}		PACKET2[8,1]	
If8h0Packet3	{...}	{...}		PACKET3[8,1]	
If8h0Packet4	{...}	{...}		PACKET4[8,1]	
If8h0PacketNumber	0		Decimal	DINT	
If8h0PassThruQryMsg	{...}	{...}		MESSAGE	
If8h0PassThruQryRX	{...}	{...}	Decimal	SINT[260]	
If8h0PassThruQryTX	{...}	{...}	Decimal	SINT[6]	
If8h0PassThruReqMsg	{...}	{...}		MESSAGE	
If8h0PassThruReqRX	{...}	{...}	Decimal	SINT[6]	
If8h0PassThruReqTX	{...}	{...}	Decimal	SINT[260]	
Local:1:C	{...}	{...}		AB:1756_MODULE:C:0	
Local:1:I	{...}	{...}		AB:1756_MODULE_DINT_96Bytes:I:0	
Local:2:C	{...}	{...}		AB:1756_MODULE:C:0	
Local:2:I	{...}	{...}		AB:1756_MODULE_DINT_96Bytes:I:0	
Local:2:O	{...}	{...}		AB:1756_MODULE_DINT_32Bytes:O:0	
Of8h0ChannelNumber	7		Decimal	DINT	
Of8h0Configuration	{...}	{...}		Of8hConfigurationBlock	
Of8h0Input	{...}	{...}		Of8hInputBlock	
Of8h0Output	{...}	{...}		Of8hOutputBlock	
Of8h0Packet0	{...}	{...}		PACKET0[8,1]	
Of8h0Packet1	{...}	{...}		PACKET1[8,1]	
Of8h0Packet2	{...}	{...}		PACKET2[8,1]	
Of8h0Packet3	{...}	{...}		PACKET3[8,1]	
Of8h0Packet4	{...}	{...}		PACKET4[8,1]	
Of8h0PacketNumber	0		Decimal	DINT	
Of8h0PassThruQryMsg	{...}	{...}		MESSAGE	
Of8h0PassThruQryRX	{...}	{...}	Decimal	SINT[260]	
Of8h0PassThruQryTX	{...}	{...}	Decimal	SINT[6]	
Of8h0PassThruReqMsg	{...}	{...}		MESSAGE	
Of8h0PassThruReqRX	{...}	{...}	Decimal	SINT[6]	
Of8h0PassThruReqTX	{...}	{...}	Decimal	SINT[260]	

Click on the tag name of the data structure you want to see

Of8h0Configuration	{...}	{...}		Of8hConfigurationBlock
Of8h0Configuration.ConfigRevision	16#00		Hex	SINT
Of8h0Configuration.ProgToFaultEn	0		Decimal	SINT
Of8h0Configuration.Spare	0		Decimal	INT
Of8h0Configuration.Ch0ConfigBits	2#0000_0010_0000_0000		Binary	INT
Of8h0Configuration.Ch0OutputRange	2		Decimal	INT
Of8h0Configuration.Ch0MaxRampRate	0.0		Float	REAL
Of8h0Configuration.Ch0FaultValue	0.0		Float	REAL
Of8h0Configuration.Ch0IdleValue	0.0		Float	REAL
Of8h0Configuration.Ch0LowSignal	4.0		Float	REAL
Of8h0Configuration.Ch0HighSignal	20.0		Float	REAL
Of8h0Configuration.Ch0LowEngineering	4.0		Float	REAL
Of8h0Configuration.Ch0HighEngineering	20.0		Float	REAL
Of8h0Configuration.Ch0LowClamp	4.0		Float	REAL
Of8h0Configuration.Ch0HighClamp	20.0		Float	REAL
Of8h0Configuration.Ch0CalBias	0.0		Float	REAL
Of8h0Configuration.Ch0Slot0Code	0		Decimal	SINT
Of8h0Configuration.Ch0Slot1Code	0		Decimal	SINT
Of8h0Configuration.Ch0Slot2Code	0		Decimal	SINT
Of8h0Configuration.Ch0Slot3Code	0		Decimal	SINT
Of8h0Configuration.Ch1ConfigBits	2#0000_0010_0000_0000		Binary	INT
Of8h0Configuration.Ch1OutputRange	2		Decimal	INT
Of8h0Configuration.Ch1MaxRampRate	0.0		Float	REAL
Of8h0Configuration.Ch1FaultValue	0.0		Float	REAL
Of8h0Configuration.Ch1IdleValue	0.0		Float	REAL
Of8h0Configuration.Ch1LowSignal	4.0		Float	REAL
Of8h0Configuration.Ch1HighSignal	20.0		Float	REAL
Of8h0Configuration.Ch1LowEngineering	4.0		Float	REAL
Of8h0Configuration.Ch1HighEngineering	20.0		Float	REAL

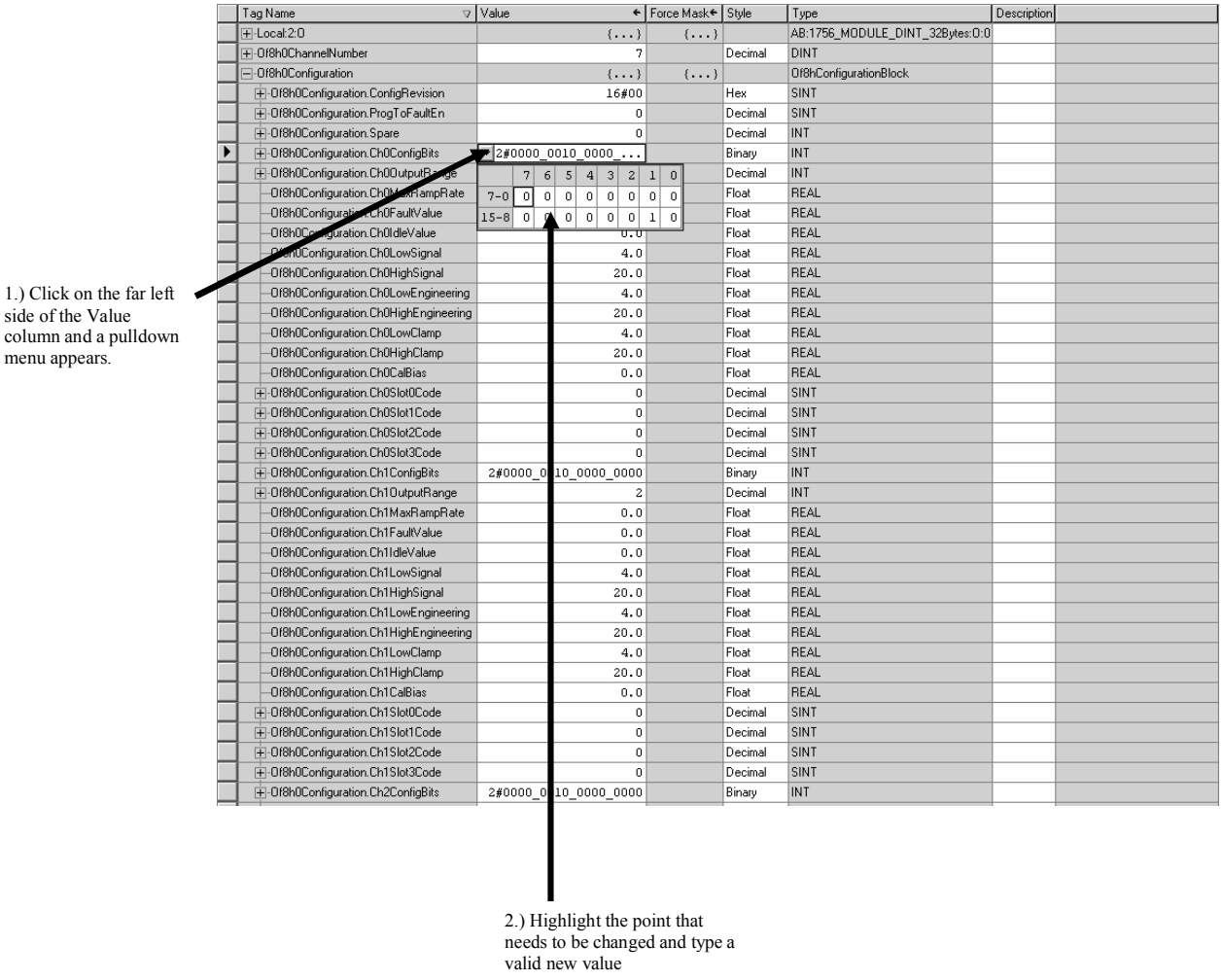
Configuration information is listed for each channel on the module

Changing Configuration Information at the Tags

There are two ways to change the configuration:

- Use a pulldown menu
- Highlight the value of a particular feature for a particular point and type a new value

Pulldown menu

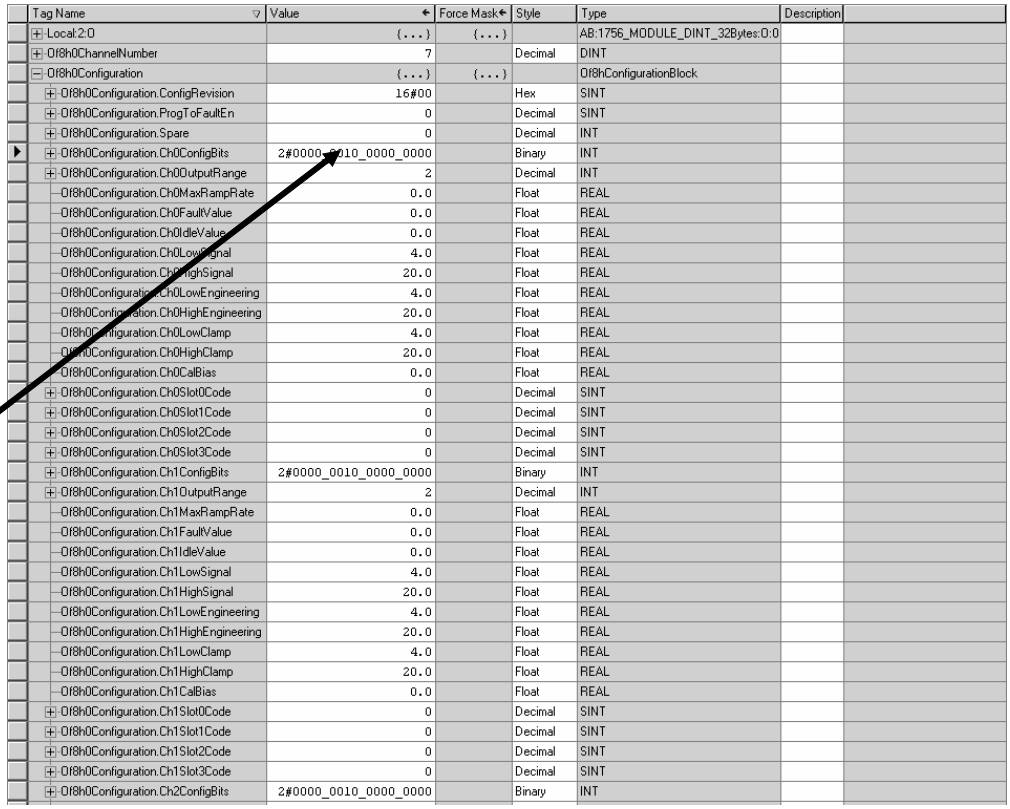


Tag Name	Value	Force Mask	Style	Type	Description
Local:2:O	{...}	{...}		AB:1756_MODULE_DINT_32Bytes:O:0	
Of8h0ChannelNumber	7		Decimal	DINT	
Of8h0Configuration	{...}	{...}		Of8hConfigurationBlock	
Of8h0Configuration.ConfigRevision	16#00		Hex	SINT	
Of8h0Configuration.ProgToFaultEn	0		Decimal	SINT	
Of8h0Configuration.Spare	0		Decimal	INT	
Of8h0Configuration.Ch0ConfigBits	2#0000_0010_0000_...		Binary	INT	
Of8h0Configuration.Ch0OutputRange			Decimal	INT	
Of8h0Configuration.Ch0MaxRampRate			Float	REAL	
Of8h0Configuration.Ch0FaultValue			Float	REAL	
Of8h0Configuration.Ch0IdleValue			Float	REAL	
Of8h0Configuration.Ch0LowSignal	4.0		Float	REAL	
Of8h0Configuration.Ch0HighSignal	20.0		Float	REAL	
Of8h0Configuration.Ch0LowEngineering	4.0		Float	REAL	
Of8h0Configuration.Ch0HighEngineering	20.0		Float	REAL	
Of8h0Configuration.Ch0LowClamp	4.0		Float	REAL	
Of8h0Configuration.Ch0HighClamp	20.0		Float	REAL	
Of8h0Configuration.Ch0CalBias	0.0		Float	REAL	
Of8h0Configuration.Ch0Slot0Code	0		Decimal	SINT	
Of8h0Configuration.Ch0Slot1Code	0		Decimal	SINT	
Of8h0Configuration.Ch0Slot2Code	0		Decimal	SINT	
Of8h0Configuration.Ch0Slot3Code	0		Decimal	SINT	
Of8h0Configuration.Ch1ConfigBits	2#0000_0010_0000_0000		Binary	INT	
Of8h0Configuration.Ch1OutputRange	2		Decimal	INT	
Of8h0Configuration.Ch1MaxRampRate	0.0		Float	REAL	
Of8h0Configuration.Ch1FaultValue	0.0		Float	REAL	
Of8h0Configuration.Ch1IdleValue	0.0		Float	REAL	
Of8h0Configuration.Ch1LowSignal	4.0		Float	REAL	
Of8h0Configuration.Ch1HighSignal	20.0		Float	REAL	
Of8h0Configuration.Ch1LowEngineering	4.0		Float	REAL	
Of8h0Configuration.Ch1HighEngineering	20.0		Float	REAL	
Of8h0Configuration.Ch1LowClamp	4.0		Float	REAL	
Of8h0Configuration.Ch1HighClamp	20.0		Float	REAL	
Of8h0Configuration.Ch1CalBias	0.0		Float	REAL	
Of8h0Configuration.Ch1Slot0Code	0		Decimal	SINT	
Of8h0Configuration.Ch1Slot1Code	0		Decimal	SINT	
Of8h0Configuration.Ch1Slot2Code	0		Decimal	SINT	
Of8h0Configuration.Ch1Slot3Code	0		Decimal	SINT	
Of8h0Configuration.Ch2ConfigBits	2#0000_0010_0000_0000		Binary	INT	

	7	6	5	4	3	2	1	0
7-0	0	0	0	0	0	0	0	0
15-8	0	0	0	0	0	0	1	0

Highlight value

1.) Highlight the value of the feature you want to change
2.) **Type in the valid new value.**



Tag Name	Value	Force Mask	Style	Type	Description
Local:2:0	{...}	{...}		AB:1756_MODULE_DINT_32Bytes:O:0	
Of8h0ChannelNumber	7		Decimal	DINT	
Of8h0Configuration	{...}	{...}		Of8hConfigurationBlock	
Of8h0Configuration.ConfigRevision	16#00		Hex	SINT	
Of8h0Configuration.ProgToFaultEn	0		Decimal	SINT	
Of8h0Configuration.Spare	0		Decimal	INT	
Of8h0Configuration.Ch0ConfigBits	2#0000_0010_0000_0000		Binary	INT	
Of8h0Configuration.Ch0OutputRange	2		Decimal	INT	
Of8h0Configuration.Ch0MaxRampRate	0.0		Float	REAL	
Of8h0Configuration.Ch0FaultValue	0.0		Float	REAL	
Of8h0Configuration.Ch0IdleValue	0.0		Float	REAL	
Of8h0Configuration.Ch0LowSignal	4.0		Float	REAL	
Of8h0Configuration.Ch0HighSignal	20.0		Float	REAL	
Of8h0Configuration.Ch0LowEngineering	4.0		Float	REAL	
Of8h0Configuration.Ch0HighEngineering	20.0		Float	REAL	
Of8h0Configuration.Ch0LowClamp	4.0		Float	REAL	
Of8h0Configuration.Ch0HighClamp	20.0		Float	REAL	
Of8h0Configuration.Ch0CalBias	0.0		Float	REAL	
Of8h0Configuration.Ch0Slot0Code	0		Decimal	SINT	
Of8h0Configuration.Ch0Slot1Code	0		Decimal	SINT	
Of8h0Configuration.Ch0Slot2Code	0		Decimal	SINT	
Of8h0Configuration.Ch0Slot3Code	0		Decimal	SINT	
Of8h0Configuration.Ch1ConfigBits	2#0000_0010_0000_0000		Binary	INT	
Of8h0Configuration.Ch1OutputRange	2		Decimal	INT	
Of8h0Configuration.Ch1MaxRampRate	0.0		Float	REAL	
Of8h0Configuration.Ch1FaultValue	0.0		Float	REAL	
Of8h0Configuration.Ch1IdleValue	0.0		Float	REAL	
Of8h0Configuration.Ch1LowSignal	4.0		Float	REAL	
Of8h0Configuration.Ch1HighSignal	20.0		Float	REAL	
Of8h0Configuration.Ch1LowEngineering	4.0		Float	REAL	
Of8h0Configuration.Ch1HighEngineering	20.0		Float	REAL	
Of8h0Configuration.Ch1LowClamp	4.0		Float	REAL	
Of8h0Configuration.Ch1HighClamp	20.0		Float	REAL	
Of8h0Configuration.Ch1CalBias	0.0		Float	REAL	
Of8h0Configuration.Ch1Slot0Code	0		Decimal	SINT	
Of8h0Configuration.Ch1Slot1Code	0		Decimal	SINT	
Of8h0Configuration.Ch1Slot2Code	0		Decimal	SINT	
Of8h0Configuration.Ch1Slot3Code	0		Decimal	SINT	
Of8h0Configuration.Ch2ConfigBits	2#0000_0010_0000_0000		Binary	INT	

Enabling and Using HART on the 1756sc-IF8H and OF8H

This chapter outlines the detailed settings and configuration related to HART communication for the 1756sc-IF8H and 1756sc-OF8H modules. These settings determine how the modules acquire HART data.

The chapter is broken down into the following sections:

- Configuring the modules for HART
- How the modules send and receive HART data
- HART protocol overview

Configuring the Modules for HART

Configuring the IF8H Module for (Hart Acquisition/Communication)

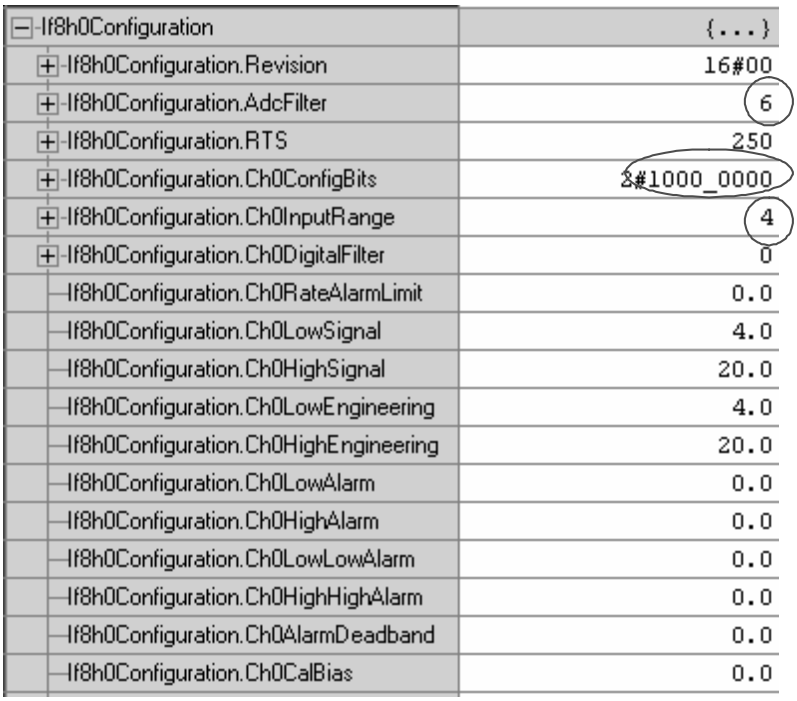
- In order for HART to be active on any given channel, the channel configuration must contain the following basic settings:
- Bit 7 (Enable HART), in the If8h0Configuration.ChXConfigBits tag, needs to be set. See chapter 5 for more details.
- An appropriate ADC filter range must be selected.

Attention: The 1000Hz ADC filter setting is not allowed when HART is enabled.

- The If8h0Configuration.ChXInputRange tag must be set to 4 for a 4 to mA current range.

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

Figure 7.1 (Channel 0 Configuration Example)



If8h0Configuration	{...}
If8h0Configuration.Revision	16#00
If8h0Configuration.AdcFilter	6
If8h0Configuration.RTS	250
If8h0Configuration.Ch0ConfigBits	2#1000_0000
If8h0Configuration.Ch0InputRange	4
If8h0Configuration.Ch0DigitalFilter	0
If8h0Configuration.Ch0RateAlarmLimit	0.0
If8h0Configuration.Ch0LowSignal	4.0
If8h0Configuration.Ch0HighSignal	20.0
If8h0Configuration.Ch0LowEngineering	4.0
If8h0Configuration.Ch0HighEngineering	20.0
If8h0Configuration.Ch0LowAlarm	0.0
If8h0Configuration.Ch0HighAlarm	0.0
If8h0Configuration.Ch0LowLowAlarm	0.0
If8h0Configuration.Ch0HighHighAlarm	0.0
If8h0Configuration.Ch0AlarmDeadband	0.0
If8h0Configuration.Ch0CalBias	0.0

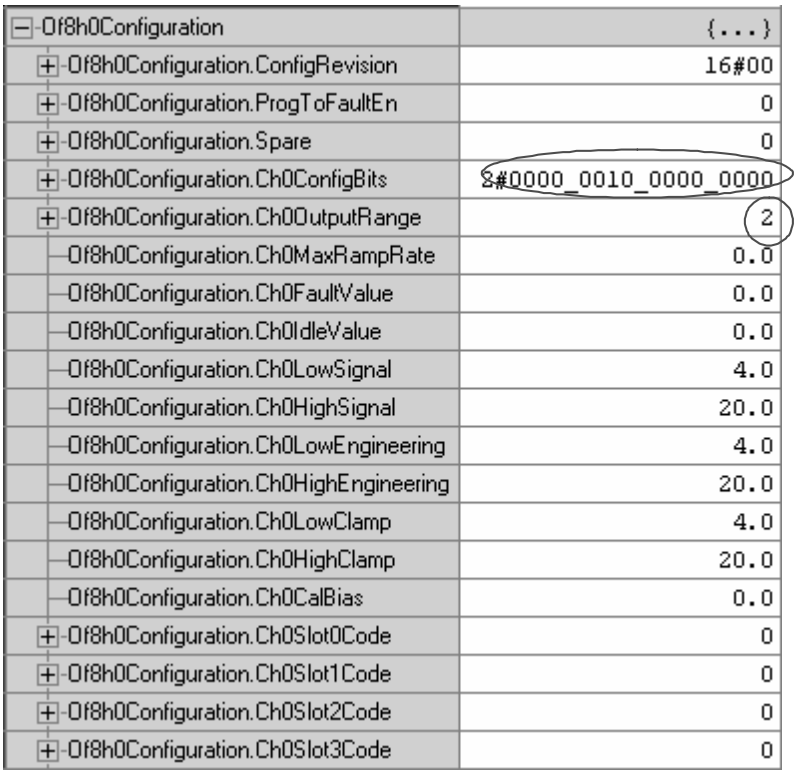
Configuring the OF8H Module for (Hart Acquisition/Communication)

In order for HART to be active on any given channel, the channel configuration must contain the following basic settings:

- Bit 9 (Enable HART), in the Of8h0Configuration.ChXConfigBits tag, needs to be set. See chapter 6 for more details.
- The Of8h0Configuration.ChXOutputRange tag must be set to 2 for a 4 to 20 mA current range.

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

Figure 7.2 (Channel 0 Configuration Example)



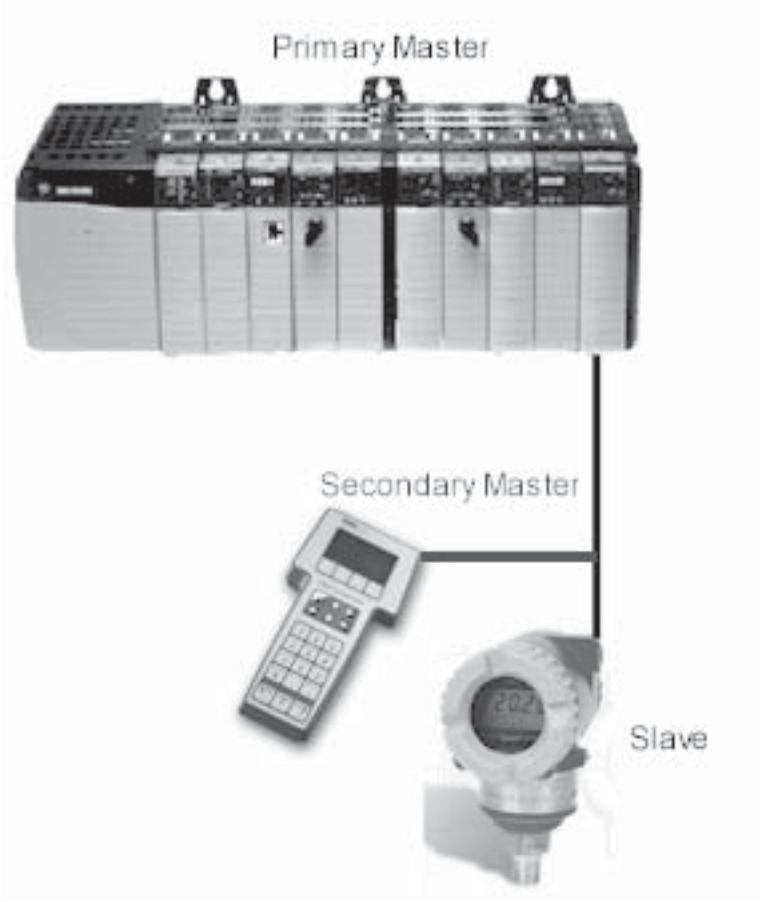
Of8h0Configuration	{...}
Of8h0Configuration.ConfigRevision	16#00
Of8h0Configuration.ProgToFaultEn	0
Of8h0Configuration.Spare	0
Of8h0Configuration.Ch0ConfigBits	8#0000_0010_0000_0000
Of8h0Configuration.Ch0OutputRange	2
Of8h0Configuration.Ch0MaxRampRate	0.0
Of8h0Configuration.Ch0FaultValue	0.0
Of8h0Configuration.Ch0IdleValue	0.0
Of8h0Configuration.Ch0LowSignal	4.0
Of8h0Configuration.Ch0HighSignal	20.0
Of8h0Configuration.Ch0LowEngineering	4.0
Of8h0Configuration.Ch0HighEngineering	20.0
Of8h0Configuration.Ch0LowClamp	4.0
Of8h0Configuration.Ch0HighClamp	20.0
Of8h0Configuration.Ch0CalBias	0.0
Of8h0Configuration.Ch0Slot0Code	0
Of8h0Configuration.Ch0Slot1Code	0
Of8h0Configuration.Ch0Slot2Code	0
Of8h0Configuration.Ch0Slot3Code	0

How the Modules Send and Receive HART Data

How the Module Connects to a Field Device

Both the HART input and Output module behave 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 3.

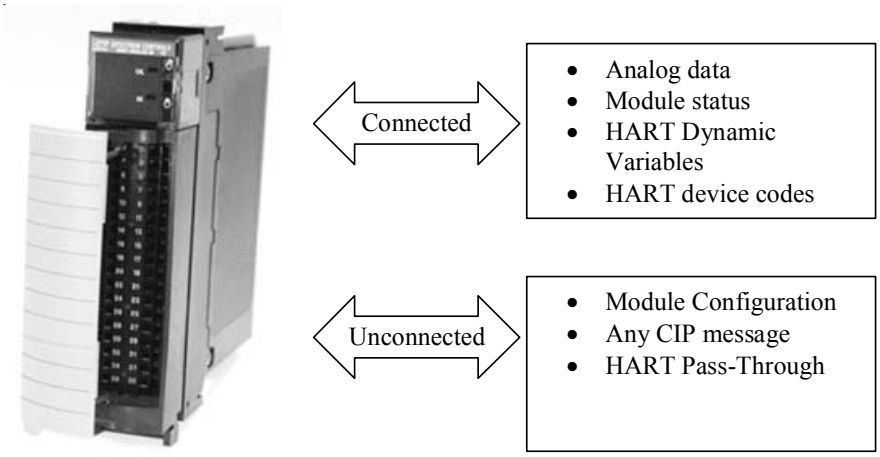
Figure 7.3 (Primary, Secondary and Slave connection)



Attention: Hart multi-drop is not supported by the IF8H or the OF8H.

HART modules communicate using two different messaging schemes connected and unconnected. Connected messages are messages that are periodically updated and maintained by the ControlLogix processor. The frequencies of the updates are adjustable and are defined by the user. Connected messages include data contained in the module's input image, and in the case of the OF8H module, the output image. Unconnected messages are messages that are transmitted over the control network and are performed asynchronously from the normal processor scan. Unconnected messages include data contained in the module's configuration image (i.e. Local:e:C), and CIP messages such as the module specific commands. Module specific commands include the HART pass-through commands, HART suspension and resume, and the get HART device information command.

Figure 7.4 (Connected and Unconnected messaging)

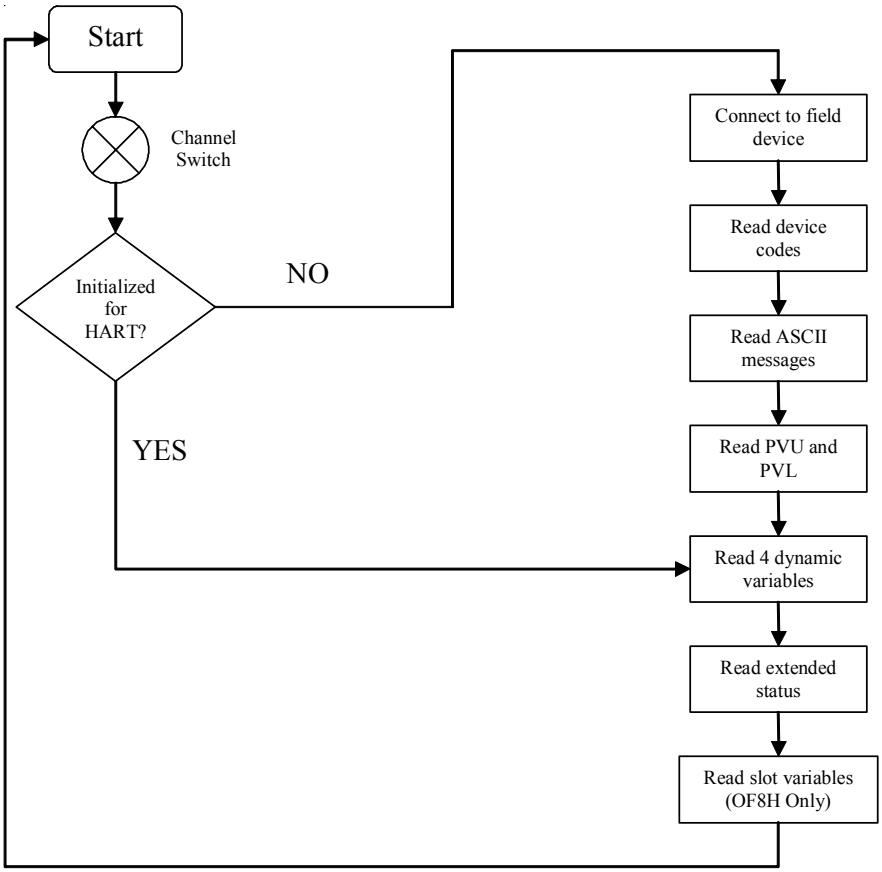


The module uses the two communication schemes to gather HART data. Gathering HART data is accomplished using two processes **auto acquisition**, and or using the **module specific commands**.

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 (i.e. Manufacturer ID, serial number, etc.), the four dynamic variables, extended device status, slot variables (OF8H only), 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 and is accessible by ladder logic. The HART data will update, on average, every 6.5 seconds if all eight 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 7.5 (Auto Acquisition Flow)



The data that is collected from the process described in figure 7.5 is buffered to the modules RAM memory. Since the amount of data returned from the auto-acquisition process is extensive, utilizing the module input tags efficiently would not be practical unless the data is multiplexed. Therefore, the data is multiplexed into five separate packets and for each individual channel. The packets are defined in tables 7.1 to 7.5.

Table 7.1(Packet 0)

Tag Name	Data Type	Style	Description
(If8h0 or Of8h0)Packet0[1]	Packet0[8,1]	NA	Two dimmensional array containing packet 0 data for all 8 channels.
(If8h0 or Of8h0)Packet0[X,0][1,2]	Packet0	NA	Packet 0 data for channel X
(If8h0 or Of8h0)Packet0[X,0].HartChannelID	INT	BIN	Bits 0 to 3: Channel number (0 – 7). 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
(If8h0 or Of8h0)Packet0[X,0].ManufacturerID	SINT	DEC	HART device Manufacturer ID
(If8h0 or Of8h0)Packet0[X,0].DeviceType	SINT	DEC	HART device type code
(If8h0 or Of8h0)Packet0[X,0].NumPreambles	SINT	DEC	Minimum number of preambles the device requires.
(If8h0 or Of8h0)Packet0[X,0].UniversalCmdCode	SINT	DEC	HART Universal command code
(If8h0 or Of8h0)Packet0[X,0].XmitterRev	SINT	DEC	HART Transmitter specific revision
(If8h0 or Of8h0)Packet0[X,0].SwRev	SINT	DEC	HART device software revision number
(If8h0 or Of8h0)Packet0[X,0].HwRev	SINT	DEC	HART device hardware revision number
(If8h0 or Of8h0)Packet0[X,0].HartFlags	SINT	BIN	HART flags
(If8h0 or Of8h0)Packet0[X,0].RangeUnits	SINT	DEC	Units code for range parameter
(If8h0 or Of8h0)Packet0[X,0].DeviceSerialNumber	SINT[3]	HEX	HART device ID number
(If8h0 or Of8h0)Packet0[X,0].DeviceTag	SINT[8]	ASCII	8 character device tag
(If8h0 or Of8h0)Packet0[X,0].DeviceDescriptor	SINT[16]	ASCII	

[1] The name in parentheses represents the module tag name (i.e. If8h0 or Of8h0) depending on which module is being used.

[2] X represents the module channel number (0 to 7)

Table 7.2(Packet 1)

Tag Name	Data Type	Style	Description
(If8h0 or Of8h0)Packet1[1]	Packet1[8,1]	NA	Two dimensional array containing packet 1 data for all 8 channels.
(If8h0 or Of8h0)Packet1[X,0][1,2]	Packet1	NA	Packet 1 data for channel X
(If8h0 or Of8h0)Packet1[X,0].HartChannelID	INT	BIN	Bits 0 to 3: Channel number (0 – 7). 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
(If8h0 or Of8h0)Packet1[X,0].HartCommStatus	SINT	BIN	HART communication status byte. Refer to appendix D for more details.
(If8h0 or Of8h0)Packet1[X,0].HartDevStatus	SINT	BIN	HART device status byte. Refer to appendix D for more details.
(If8h0 or Of8h0)Packet1[X,0].HartPV	REAL	FLOAT	HART Primary Variable
(If8h0 or Of8h0)Packet1[X,0].HartSV	REAL	FLOAT	HART Secondary Variable
(If8h0 or Of8h0)Packet1[X,0].HartTV	REAL	FLOAT	HART Tertiary Variable
(If8h0 or Of8h0)Packet1[X,0].HartFV	REAL	FLOAT	HART Fourth Variable
(If8h0 or Of8h0)Packet1[X,0].HartPVUnits	SINT	DEC	HART Primary Variable units code
(If8h0 or Of8h0)Packet1[X,0].HartSVUnits	SINT	DEC	HART Secondary Variable units code
(If8h0 or Of8h0)Packet1[X,0].HartTVUnits	SINT	DEC	HART Tertiary Variable units code
(If8h0 or Of8h0)Packet1[X,0].HartFVUnits	SINT	DEC	HART Fourth Variable units code
(If8h0 or Of8h0)Packet1[X,0].PV_Assignment	SINT	DEC	HART Primary Variable code
(If8h0 or Of8h0)Packet1[X,0].SV_Assignment	SINT	DEC	HART Secondary Variable code
(If8h0 or Of8h0)Packet1[X,0].TV_Assignment	SINT	DEC	HART Tertiary Variable code
(If8h0 or Of8h0)Packet1[X,0].FV_Assignment	SINT	DEC	HART Fourth Variable code
(If8h0 or Of8h0)Packet1[X,0].RangeLow	REAL	FLOAT	Low transmitter range for analog signal in engineering units
(If8h0 or Of8h0)Packet1[X,0].RangeHi	REAL	FLOAT	High transmitter range for analog signal in engineering units
(If8h0 or Of8h0)Packet1[X,0].Pad	SINT[4]	DEC	Packet pad (32 bit alignment)

[1] The name in parentheses represents the module tag name (i.e. If8h0 or Of8h0) depending on which module is being used.

[2] X represents the module channel number (0 to 7)

Table 7.3 (Packet 2)

Tag Name	Data Type	Style	Description
(If8h0 or Of8h0)Packet2[1]	Packet2[8,1]	NA	Two dimensional array containing packet 2 data for all 8 channels.
(If8h0 or Of8h0)Packet2[X,0][1,2]	Packet2	NA	Packet 2 data for channel X
(If8h0 or Of8h0)Packet2[X,0].HartChannelID	INT	BIN	Bits 0 to 3: Channel number (0 – 7). 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
(If8h0 or Of8h0)Packet2[X,0].Slot0Data	REAL	Float	Variable for slot 0
(If8h0 or Of8h0)Packet2[X,0].Slot1Data	REAL	Float	Variable for slot 1
(If8h0 or Of8h0)Packet2[X,0].Slot2Data	REAL	Float	Variable for slot 2
(If8h0 or Of8h0)Packet2[X,0].Slot3Data	REAL	Float	Variable for slot 3
(If8h0 or Of8h0)Packet2[X,0].Slot0Units	SINT	DEC	Slot 0 units code
(If8h0 or Of8h0)Packet2[X,0].Slot1Units	SINT	DEC	Slot 1 units code
(If8h0 or Of8h0)Packet2[X,0].Slot2Units	SINT	DEC	Slot 2 units code
(If8h0 or Of8h0)Packet2[X,0].Slot3Units	SINT	DEC	Slot 3 units code
(If8h0 or Of8h0)Packet2[X,0].Slot0Assignment	SINT	DEC	Slot 0 variable code
(If8h0 or Of8h0)Packet2[X,0].Slot1Assignment	SINT	DEC	Slot 1 variable code
(If8h0 or Of8h0)Packet2[X,0].Slot2Assignment	SINT	DEC	Slot 2 variable code
(If8h0 or Of8h0)Packet2[X,0].Slot3Assignment	SINT	DEC	Slot 3 variable code
(If8h0 or Of8h0)Packet2[X,0].Pad	SINT[12]	DEC	Packet pad

[1] The name in parentheses represents the module tag name (i.e. If8h0 or Of8h0) depending on which module is being used.

[2] X represents the module channel number (0 to 7)

Attention: Slot variables are not acquired automatically by the input module. Therefore, packet 2 is skipped by the auto-acquisition process.

Table 7.4 (Packet 3)

Tag Name	Data Type	Style	Description
(If8h0 or Of8h0)Packet3[1]	Packet3[8,1]	NA	Two dimensional array containing packet 3 data for all 8 channels.
(If8h0 or Of8h0)Packet3[X,0][1,2]	Packet3	NA	Packet 3 data for channel X
(If8h0 or Of8h0)Packet3[X,0].HartChannelID	INT	BIN	Bits 0 to 3: Channel number (0 – 7). 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
Message	SINT[32]	ASCII	32 character message
Pad	SINT[4]	DEC	Pad 32 bit alignment.

[1] The name in parentheses represents the module tag name (i.e. If8h0 or Of8h0) depending on which module is being used.

[2] X represents the module channel number (0 to 7)

Table 7.5 (Packet 4)

Tag Name	Data Type	Style	Description
(If8h0 or Of8h0)Packet4[1]	Packet4[8,1]	NA	Two dimensional array containing packet 4 data for all 8 channels.
(If8h0 or Of8h0)Packet4[X,0][1,2]	Packet4[8,1]	NA	Packet 4 data for channel X
(If8h0 or Of8h0)Packet4[X,0].HartChannelID	INT	BIN	Bits 0 to 3: Channel number (0 – 7). 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
Date	SINT[3]	DEC	Stored date in the field device
FinalAssemblyNumber	SINT[3]	DEC	The final assembly number is used for identifying the materials and electronics that comprise the field device.
ExtendedStatus	SINT[24]	DEC	The extended status returned by HART command 48
Pad	SINT[3]	DEC	Pad 32 bit alignment

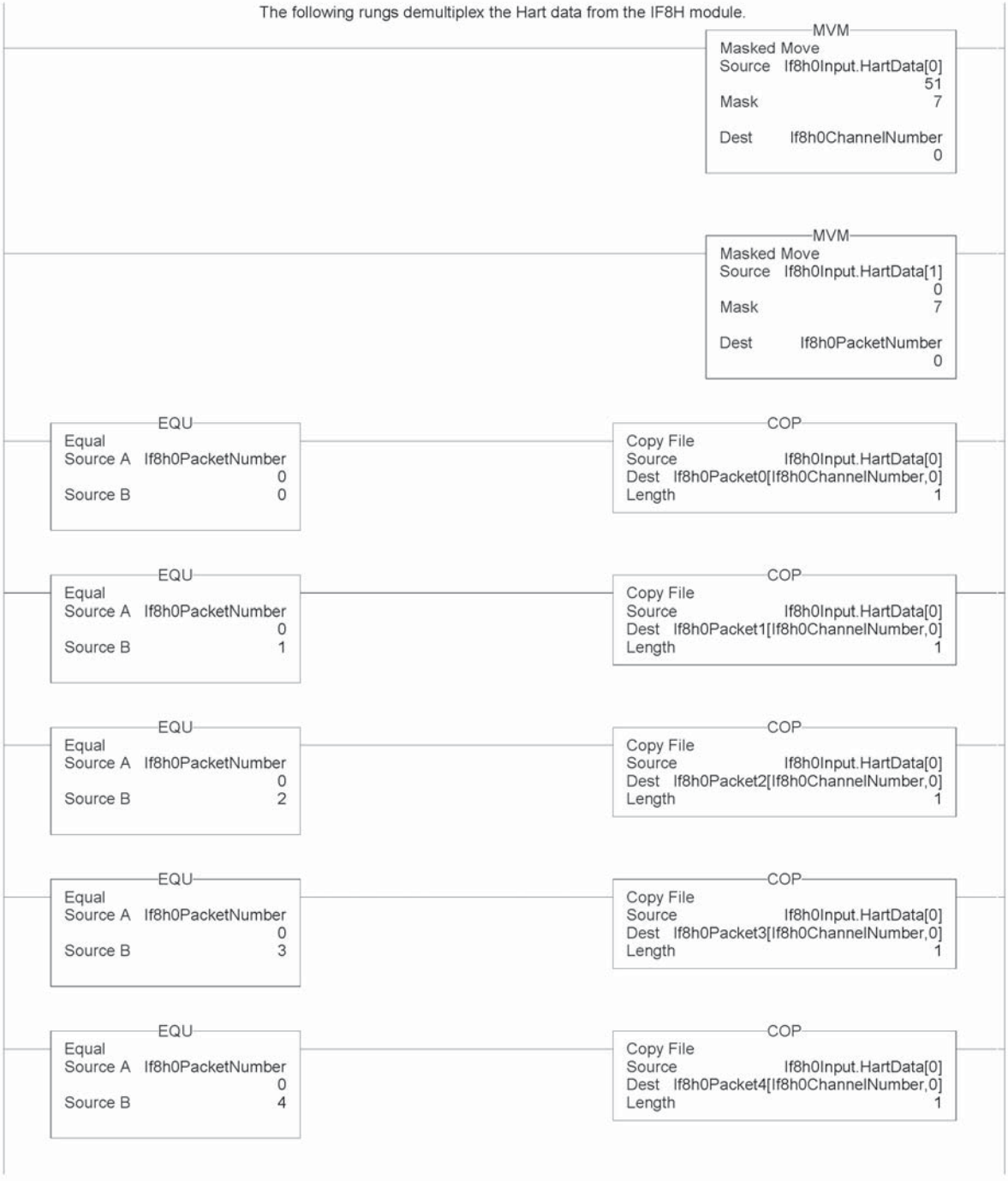
[1] The name in parentheses represents the module tag name (i.e. If8h0 or Of8h0) depending on which module is being used.

[2] X represents the module channel number (0 to 7)

Note: Not all of the data that is returned by the process outlined in figure 7.5 gets passed to the packets. In order 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 HART data acquired by the auto acquisition process is copied to the packet tags by using ladder. The ladder simply copies the data stored in the (If8h0 or Of8h0)Input.HartDatainput tag., which is multiplexed for the appropriate packet depending on the state of bits 0, 1, 2 and 8, 9, 10, found in the first two bytes of the tag. Bits 0, 1, 2 determine the current channel that is being scanned and bits 8, 9, and 10 determine the appropriate packet. The ladder example, shown in figure 7.6, buffers the data from the HartDataInput tag to the correct packets for the IF8H module.

Figure 7.6 (Demultiplexing Ladder)

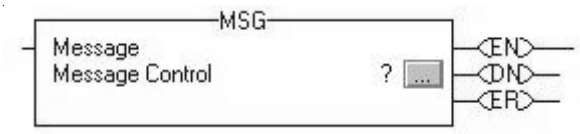


Note: The ladder in figure 7.6 can be found in the project sample file located on our website at (www.spectrumcontrols.com)

Module Specific Commands

Both the HART input and output modules provide module specific commands. The commands are passed to the module using unconnected messaging. Unconnected messaging simply refers to the ability to communicate to the module, over a control network (i.e. CNET, Ethernet, Etc.), without the need of the owner PLC. However, an unconnected message can be sent from a PLC to the HART module using the MSG ladder instruction. See figure below.

Figure 7.7 (Message Instruction)



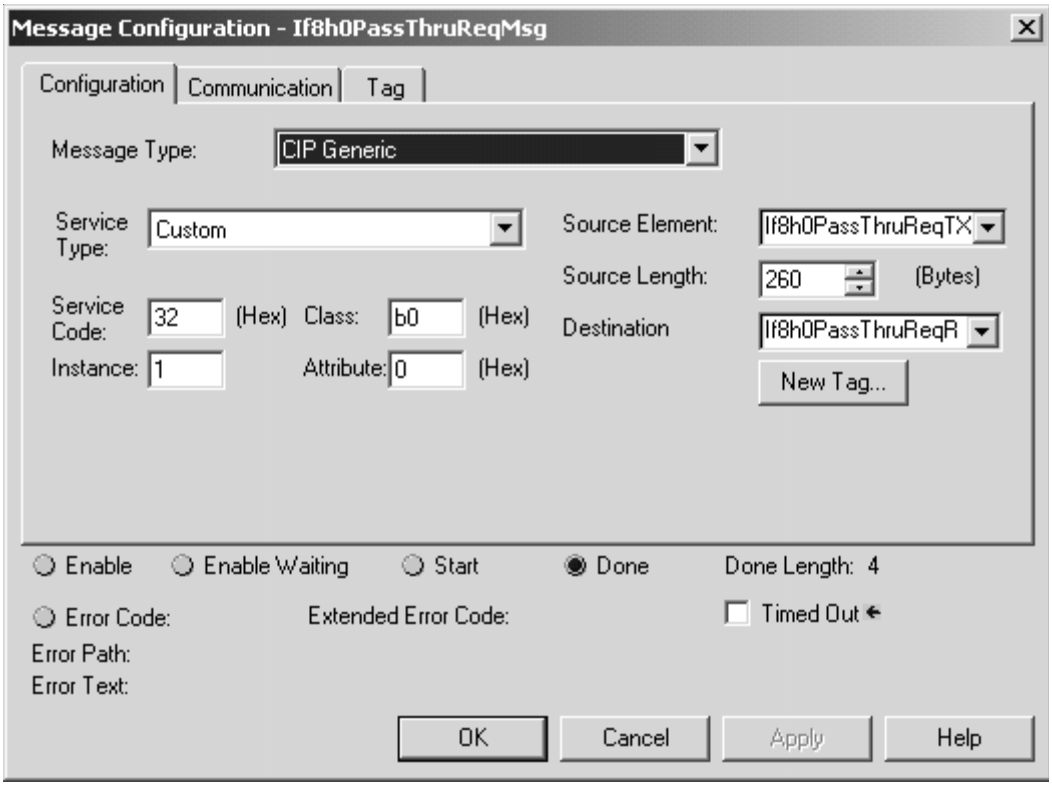
The MSG instruction must be configured correctly to establish communication to the HART module. Each module specific command uses the same generic CIP message configuration. The generic CIP message configuration used by the HART input and output module is shown in the table below.

Table 7.6 (Generic CIP Configuration)

Unconnected Message Header		
Field	Value	Definition
Message Type	"CIP Generic"	Control and Information Protocol (CIP) A native communications protocol used on Allen-Bradley's ControlNet communication network.
Service Code	0x32	
Class Name	0xB0	
Instance Name	0x01	
Object Attribute	Leave blank	
Source	?	The tag defining the data sent to the HART module
Number of elements	?	The number of elements defined by the command that is used
Destination	?	The tag where the reply data will be written

The MSG configuration screen is then loaded with the data from table 7.6. See figure below.

Figure 7.8 (Message Configuration Dialog)



After the configuration screen is loaded with the data, the actual module specific commands are loaded into the MSG source tag. When the MSG instruction executes, the module specific command is passed to the module via the MSG "source" tag and the reply data is passed to the MSG "destination" tag. The module specific commands include, *HART suspension*, *HART resume*, *get device information*, and the *HART pass-through commands*. The tables on the following pages show the format for each module specific command.

Note: The source length should be large enough to accommodate the HART message string.

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 table 7.7. 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 7.7 (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**
Unconnected Message Header		
HART Channel Number	0x00 – 0x07 (1 byte)	Module input channel number for HART command
Command Number	0x03 (1 byte)	The command number to obtain HART device information

Table 7.8 (Response If Device Information Is Not Available)

HART Get Device Information - reply packet structure		
Field	**Value**	**Definition**
Unconnected Message Header		
HART Channel Number	0x00 – 0x07 (1 byte)	Module input channel number for HART command
Status	(1 byte) 34 = RUNNING 35 = 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 7.9 (Response When Device Information Is Available)

HART Get Device Information - reply packet structure		
Field	**Value**	**Definition**
Unconnected Message Header		
HART Channel Number	0x00 – 0x07 (1 byte)	Module input channel number for HART command
Status	00 = SUCCESS	Command status
Count	(1 byte)	Number of data bytes to following.
HARTManufacturerIDCode	(1 byte)	CMD#0, Byte 1
HARTDeviceTypeCode	(1 byte)	CMD#0, Byte 2
HARTPreambles	(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
HARTDeviceID	(3 bytes) Device serial number	CMD#0, Bytes 9-11
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
HARTFinalAssemblyNumber	(3 bytes)	CMD#16, Bytes 0-2
HARTMessage	(32 bytes unpacked ASCII)	CMD#12, Bytes 0-23
HARTPVCode	(1 byte)	CMD#50, Bytes 0
HARTSVCode	(1 byte)	CMD#50, Bytes 1
HARTTVCode	(1 byte)	CMD#50, Bytes 2
HARTFVCode	(1 byte)	CMD#50, Bytes 3
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
HARTFVUnits	(1 byte)	CMD#3, Byte 19 0 if not present
HARTPVLowerRange	(4 bytes – Floating Point Value)	CMD#15, Bytes 3-6
HARTPVUpperRange	(4 bytes – Floating Point Value)	CMD#15, Bytes 7-10

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 of 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 of the following conditions are met:

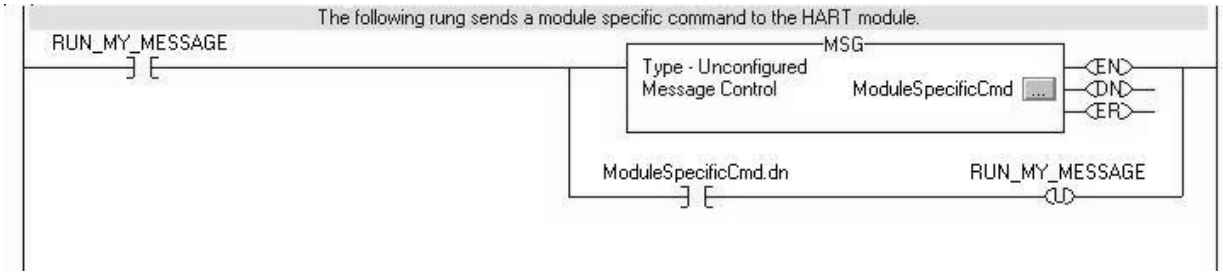
- Command and HART Channel number are both valid.
- HART channel is enabled and communication has been established, meaning at least the device addressing information is available.
- HART channel is already in the state of gathering device information. Reply will be sent back without additional events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

The following figure demonstrates how a module specific command can be sent to the module using ladder logic.

Figure 7.9 (Sending a Module Specific Command Using Ladder)



HART Channel Suspension and Resume

Sometimes referred to as "Out of Service" and "In Service" respectively, these commands can be utilized to suspend or resume operation of an enabled HART channel. When a HART Suspend Channel command is sent, the HART module will keep the current HART configuration information and stop all communication processes on the selected channel. However, there will be overriding conditions such as configuration change affecting the overall module operation, which will cause the HART function to reset based on the new configuration. Normal HART operation will resume if the HART Resume command is sent to the module during a HART Suspension.

Note: If the resume command is received, without previously receiving a suspension command, it will be ignored.

Note: The selected channel will resume normal HART operations three minutes after the Suspension command has been received by the module. Pass-through for that channel resets the timer to3 minutes.

Table 7.10 (HART Suspend and Resume Command)

HART Channel Suspend/Resume command request – command message packet structure		
Field	**Value**	**Definition**
Unconnected Message Header		
HART Channel Number	0x00 – 0x07 (1 byte)	Enabled HART channel number
Command Number	(1 byte) 0x05: Suspend 0x06: Resume	The command number to suspend or resume

Table 7.11 (HART Suspend and Resume Reply Packet)

HART Channel Suspend/Resume command request – reply packet structure		
Field	**Value**	**Definition**
Unconnected Message Header		
HART Channel Number	0x00 – 0x07 (1 byte)	Echo of the HART channel number received
Status	(1 byte) 00 = SUCCESS 35 = DEAD	Command status
Count	(1 byte)	Set to 1
Handle	0	Fill byte of zero to keep command response common among all replies.

The command status, the second byte in the reply packet for the module specific command, can return two different responses, SUCCESS, 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 number is an enabled channel
- The identified HART channel finished all of the start-up connection process.
- The I/O module will not be checking for matching set of suspend/ resume commands. This means, if already suspended, and receives another suspend, SUCCESS will be returned still. Similarly if the system is operating as normal, and receives a resume command, it will

ignore the command and continue operation. This state of operation will not be maintained after power-up or when configuration changes.

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

- Command and HART Channel number are both valid.
- HART channel is not enabled
- HART communication has not been established, meaning that the 5-byte unique address has not been determined yet, or the module is still obtaining device information.
- All other conditioned not generating SUCCESS.

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 another word, 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 the HandleTimeout (as defined in the Configuration Tag) timer. The reply message will be kept persistent during the HandleTimeout period. When the timeout occurs 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.

Attention: 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 "big-endian". Big-endian is an order in which the "big end" (most 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 "little-endian" Refer to chapter 8 for a ladder sample demonstrating the process of swapping the order of the bytes.

Table 7.12 (HART Pass-Through Command Request)

HART pass through command request – command message packet structure		
Field	**Value**	**Definition**
Unconnected Message Header		
HART Channel Number	0x00 – 0x07 (1 byte)	Module input channel number for HART command
Command Number	0x01 (1 byte)	The command number to issue a HART pass-through command.
HART Command	N bytes N = Length of message – 2 Contents are as follows: Start or Delimiter (1 byte): 0x82 Long form Address (5 bytes) 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 7.13 (HART Pass-Through Command Request Reply)

HART pass through command request – reply packet structure		
Field	**Value**	**Definition**
Unconnected Message Header		
HART Channel Number	0x00 – 0x07 (1 byte)	Module input channel number for HART command
Status	(1 byte) 33 = INITIATE 35 = DEAD (bad request)	Command status
Count	(1 byte)	Set to 1
Handle	(1 byte) 0 (bad when status is 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 7.14 (HART Pass-Through Command Complete Query)

HART pass through command complete query - command message packet structure		
Field	**Value**	**Definition**
Unconnected Message Header		
HART Channel Number	0x00 – 0x07 (1 byte)	Module input channel number for HART command
Command Number	0x0C (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 7.15 (HART Pass-Through Command Complete Query Reply)

HART pass through command complete query - reply packet structure		
Field	**Value**	**Definition**
Unconnected Message Header		
HART Channel Number	0x00 – 0x07 (1 byte)	Module input channel number for HART command
Status	(1 byte) 34 = RUNNING 35 = DEAD (bad request)	Command status
Count	(1 byte)	Length of Handle + HART Response Data in bytes (if Success)
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 7.16 (HART Pass-Through Command Complete Query - Reply Packet Structure)

HART pass through command complete query - reply packet structure		
Field	**Value**	**Definition**
Unconnected Message Header		
HART Channel Number	0x00 – 0x07 (1 byte)	Module input channel number for HART command
Status	(1 byte) 00 = SUCCESS	Command status
Count	(1 byte)	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 conditioned not generating RUNNING or SUCCESS. Examples are: 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 using ladder.

Figure 7.10a (HART Pass-Through Request and Query Process)

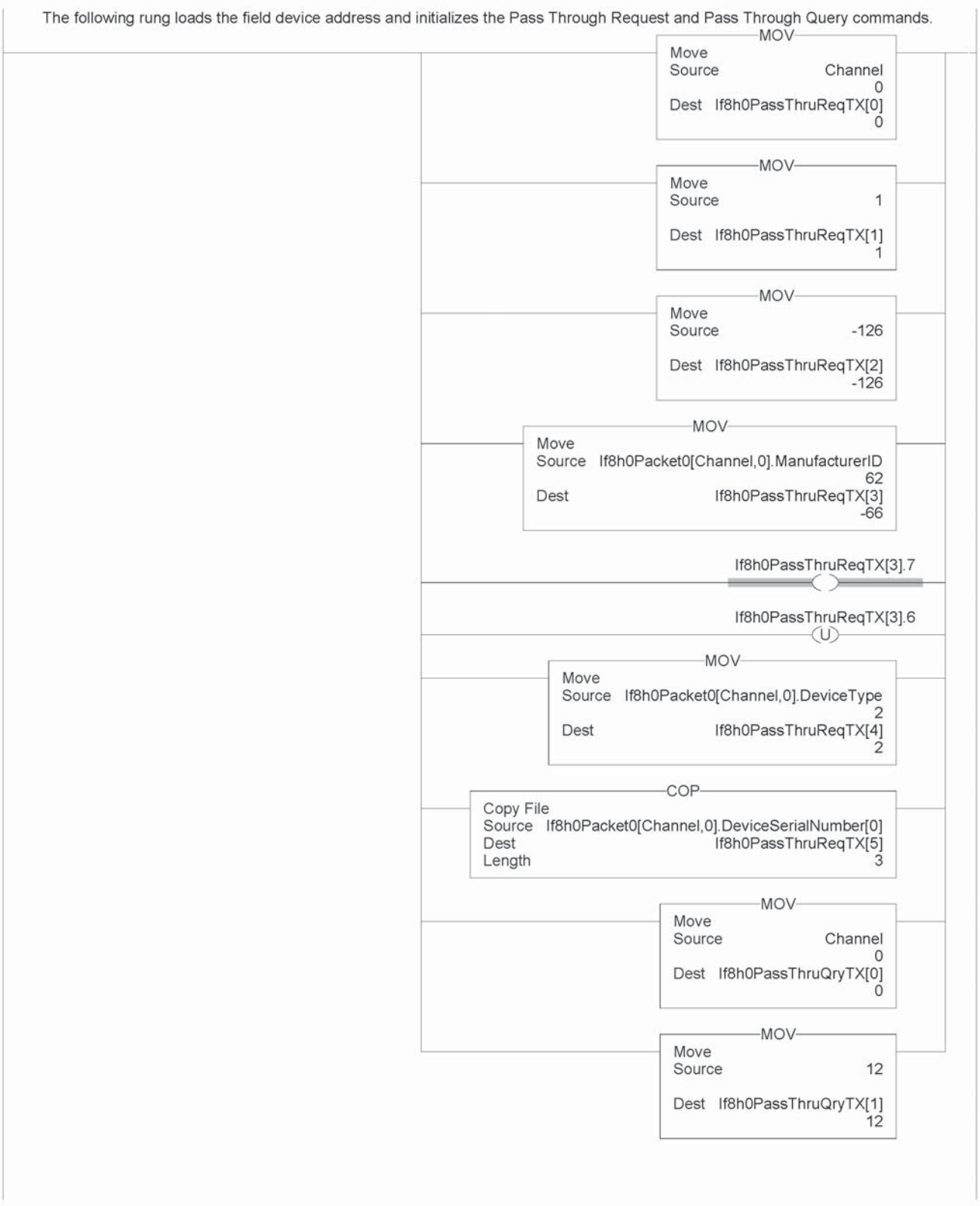


Figure 7.10b (HART Pass-Through Request and Query Process)

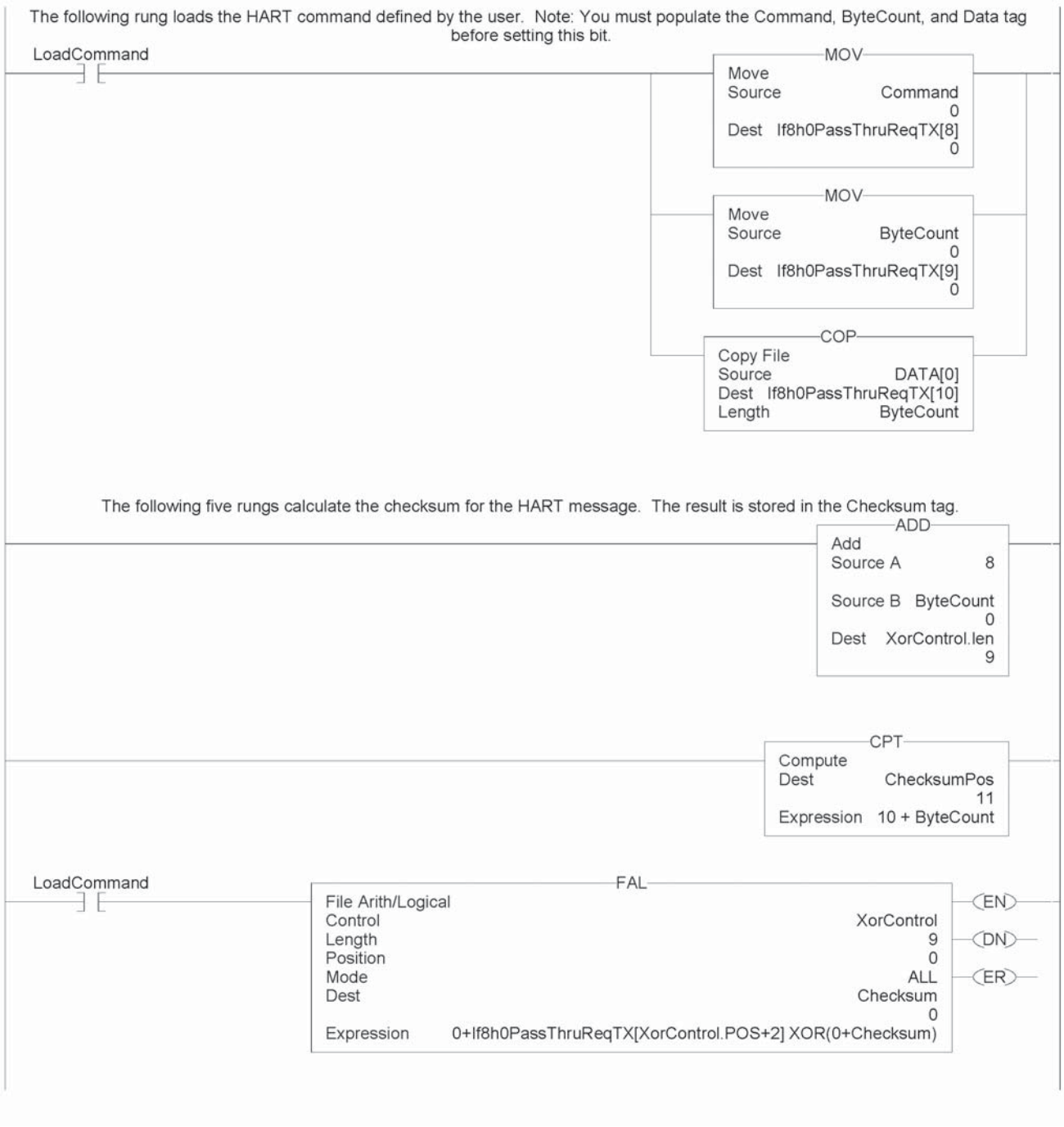


Figure 7.10c (HART Pass-Through Request and Query Process)

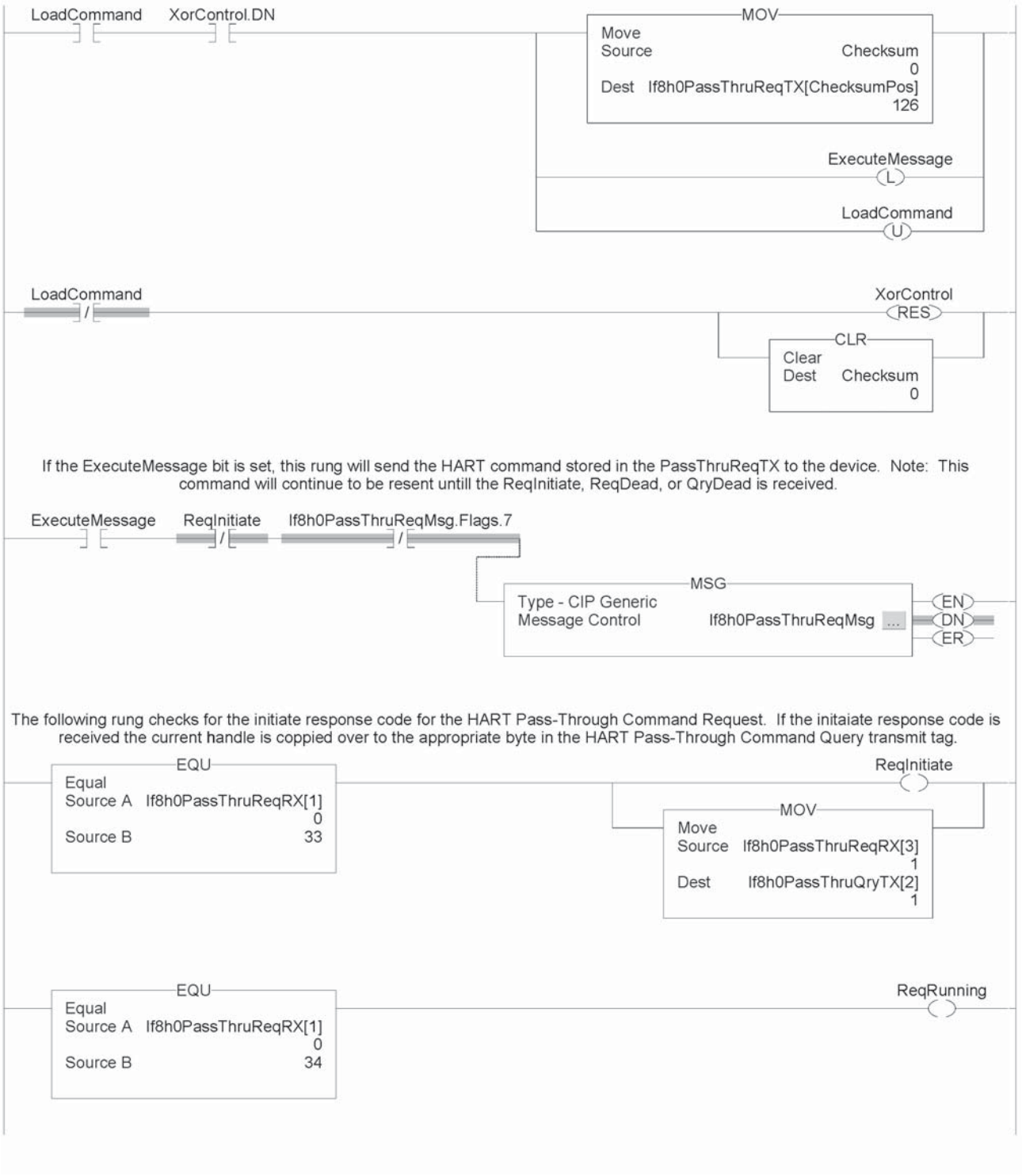


Figure 7.10d (HART Pass-Through Request and Query Process)

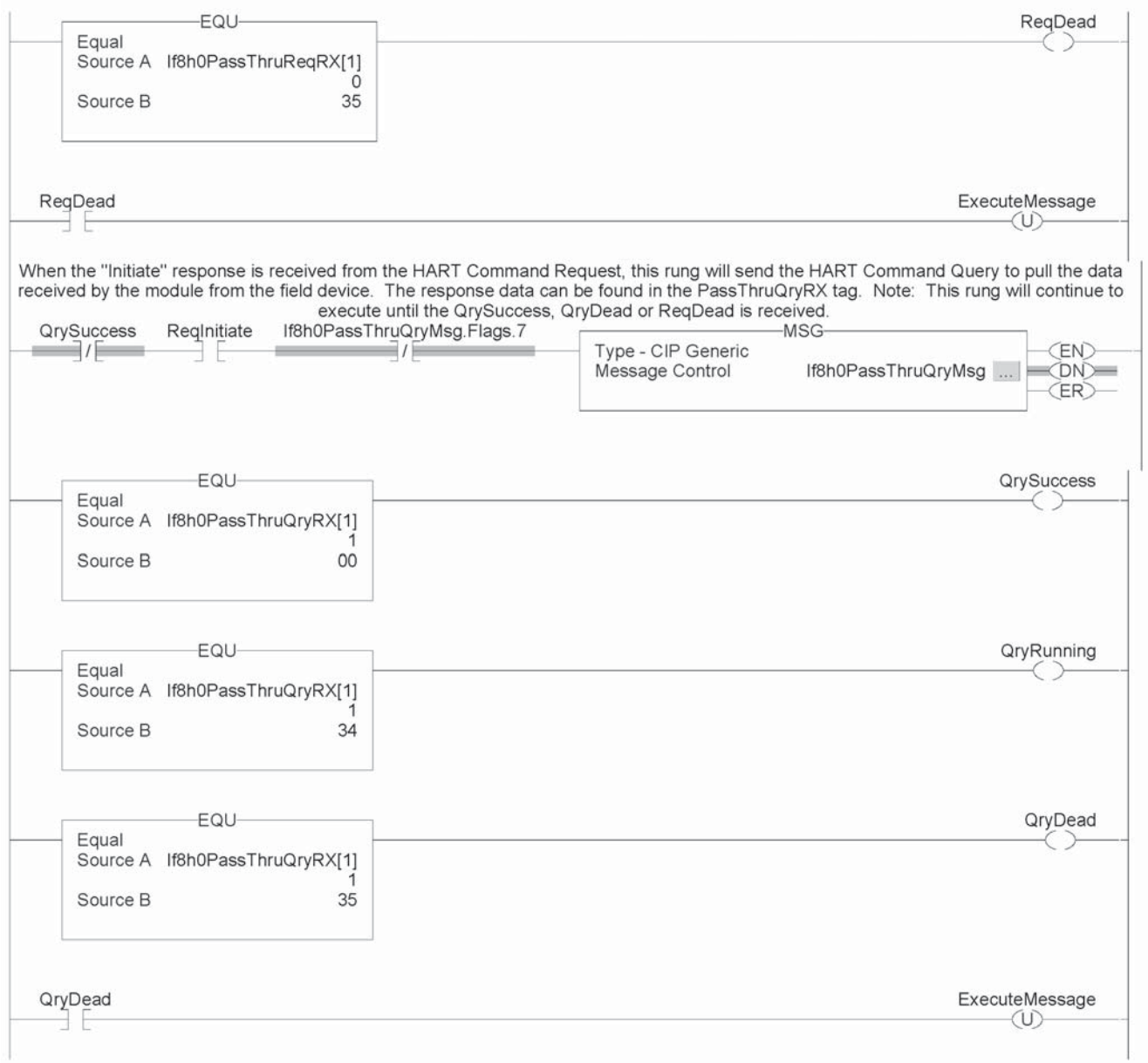
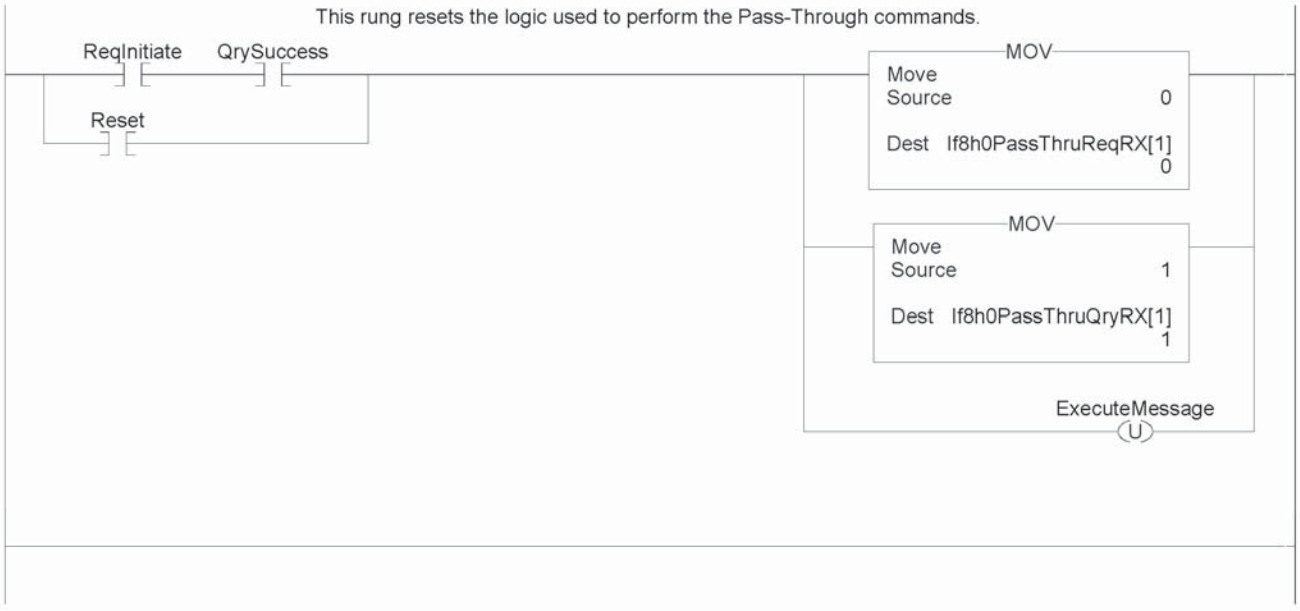


Figure 7.10e (HART Pass-Through Request and Query Process)



Note: The ladder in figure 7.10 can be found in the project sample file located on our website at (www.spectrumcontrols.com)

HART Protocol Overview

In order to read and write HART commands to and from the field device reliably using the IF8H or OF8H, you must have a basic knowledge of the HART protocol. This section will explain in detail the various pieces 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.

Message Format

HART protocol specifies a message structure as follows:

Figure 7.11 (HART Message Structure)

PREAMBLE	START CHARCTER	ADDRESS	COMMAND	BYTE COUNT	STATUS	DATA	CHECKSUM

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

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 4-20mA 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 IF8H and the OF8H support only the long frame format.

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.

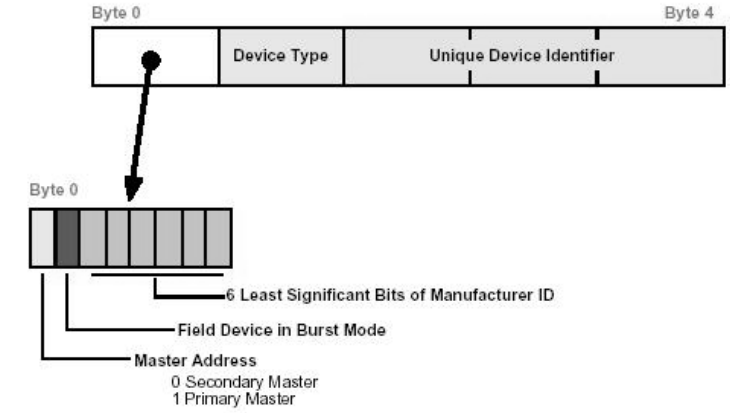
Table 7.17 (Start Character Definition)

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

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 IF8H and OF8H, use address 1, and secondary masters, such as handhelds, use address 0. Please see figure below.

Figure 7.18 (Long Frame Address)



Note: The IF8H and OF8H do 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 IF8H and OF8H automatically pull 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 2.

Command

The command byte contains an integer (0 to hex FF or decimal 255) 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 particular function is provided; and device specific commands, which are for functions more or less unique to a particular 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 D.

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.

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

Now that you're familiar with the bits and pieces that make up a HART message, the next step will be 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 that is used in the pass-through MSG block. For the example below, we will use the tag (If8h0 or Of8h0)PassThruReqTX, which was used in the ladder sample shown in figure 7.

Figure 7.19

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 charcter
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

The HART message string, shown in figure 19, performs HART command 35 (write range values). Once the tags are populated with the HART message, the message can be sent using the ladder in figure 7. The reply for the HART command will be found in the (If8h0 or Of8ho)PassThruQryRX tag. The response message should look like the table shown below.

Figure 7.20

Tag Name	Value in Hex	Description
HART_PASS_THRU_QRY_RX[0]	00	HART channel
HART_PASS_THRU_QRY_RX[1]	01	Pass-through command designator
HART_PASS_THRU_QRY_RX[2]	15	Length of handle + HART response data
HART_PASS_THRU_QRY_RX[3]	02	Message handle
HART_PASS_THRU_QRY_RX[4]	86	Start charcter
HART_PASS_THRU_QRY_RX[5]	BE	Long address byte 0
HART_PASS_THRU_QRY_RX[6]	02	Long address byte 1
HART_PASS_THRU_QRY_RX[7]	0C	Long address byte 2
HART_PASS_THRU_QRY_RX[8]	77	Long address byte 3
HART_PASS_THRU_QRY_RX[9]	37	Long address byte 4
HART_PASS_THRU_QRY_RX[10]	23	HART command = 35 decimal
HART_PASS_THRU_QRY_RX[11]	0B	Byte count = 11 decimal
HART_PASS_THRU_QRY_RX[12]	00	Status Byte 0
HART_PASS_THRU_QRY_RX[13]	00	Status Byte 1
HART_PASS_THRU_QRY_RX[14]	20	Range units code = 32 decimal
HART_PASS_THRU_QRY_RX[15]	44	Upper Range value (This is a floating point value = 600) Note: The bytes are in reverse order.
HART_PASS_THRU_QRY_RX[16]	16	
HART_PASS_THRU_QRY_RX[17]	00	
HART_PASS_THRU_QRY_RX[18]	00	
HART_PASS_THRU_QRY_RX[19]	C3	Lower Range value (This is a floating point value = -150) Note: The bytes are in reverse order.
HART_PASS_THRU_QRY_RX[20]	16	
HART_PASS_THRU_QRY_RX[21]	00	
HART_PASS_THRU_QRY_RX[22]	00	
HART_PASS_THRU_QRY_RX[23]	F9	Checksum

Programming Examples

Earlier chapters explained how the tag configuration defines the way the module operates. This chapter shows some basic programming which controls the operation of the module. It also provides you with segments of ladder logic specific to unique situations that might apply to your programming requirements.

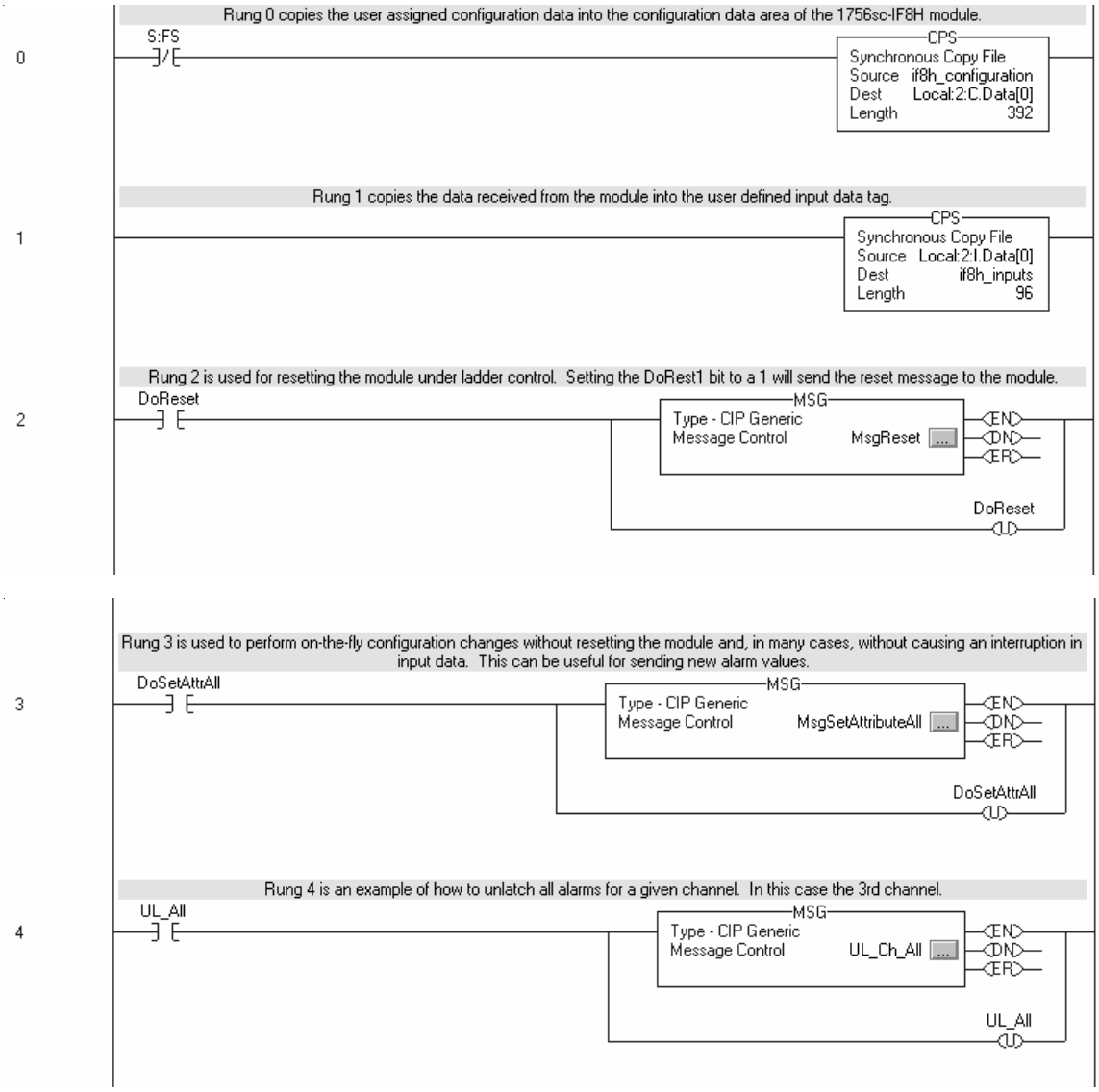
Initial Programming

Figure 8.1 illustrates some basic ladder logic commands which will allow you to:

- program the initial configuration into the module
- copy data to user defined tags
- reset the module
- make on-the-fly configuration changes
- unlatch alarms

Additional ladder logic and configuration samples may also be found on our web site: **www.spectrumcontrols.com**.

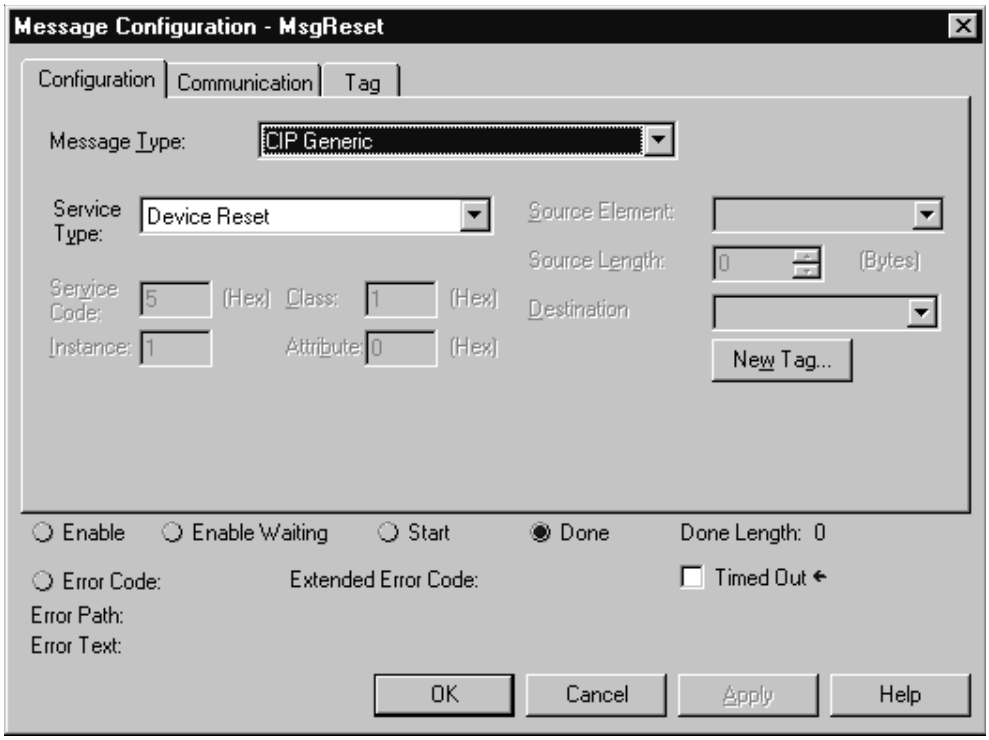
Figure 8.1 (Sample Ladder Logic)



Rung 0 - This rung copies the configuration data (IF8H_Config) into the module's configuration image memory. This rung is required.

Rung 1 - This rung copies the input data received from the module's input memory into the IF8H_Input tag for monitoring and ladder usage. this rung is required.

Rung 2 - This is an optional example rung indicating how to reset the module via ladder logic.

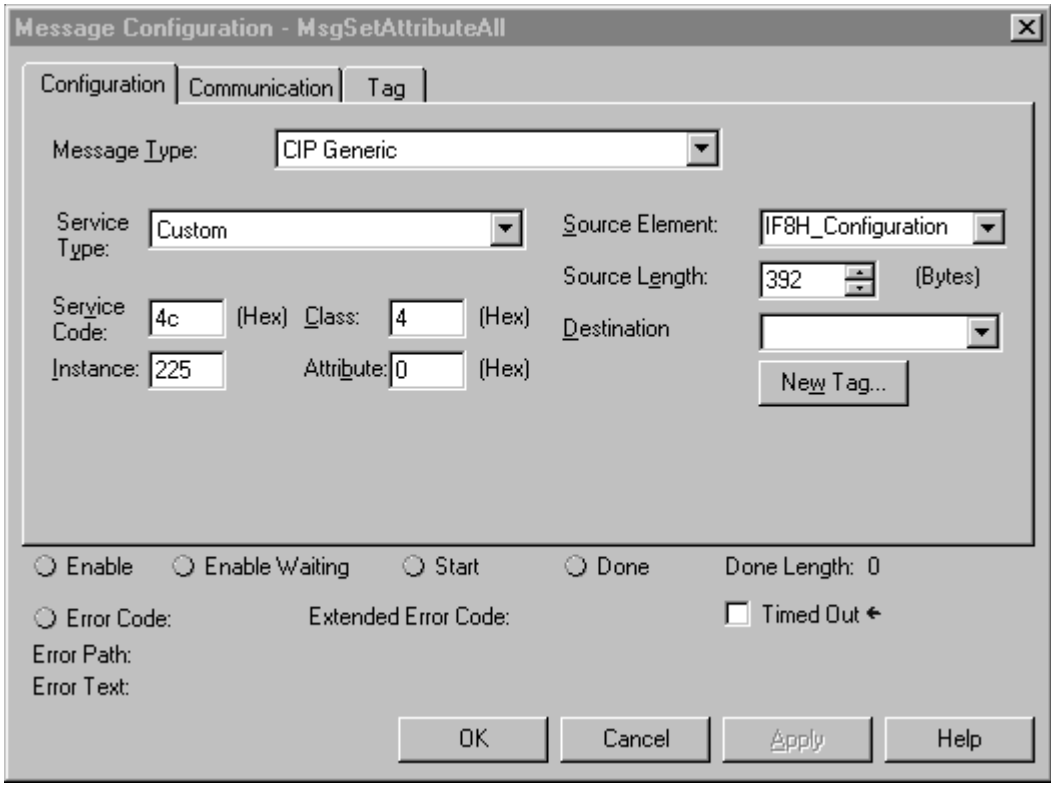


Rung 3 - This is an optional example rung indicating how to send on-the-fly configuration data to the module. This is useful if you would like to change channel alarm or scaling tags without causing interruption in channel updates. Changing other tags will cause a 2.5 second delay in channel updates but the connection will not be interrupted.

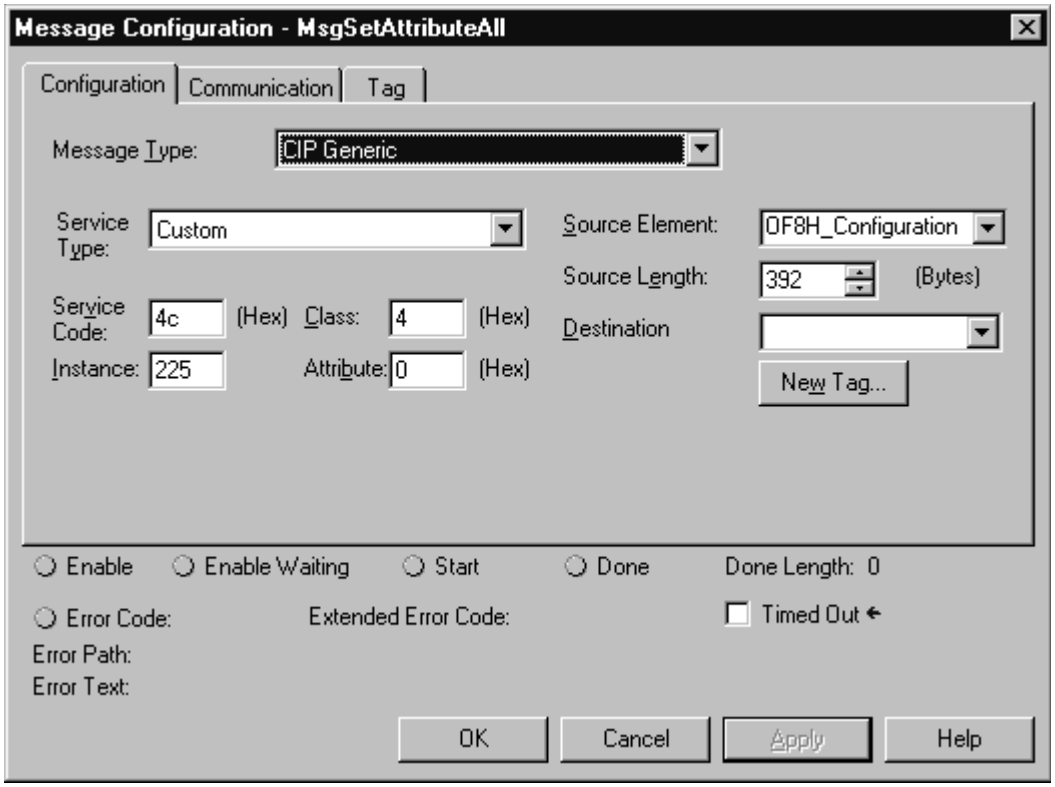
Continued on next page...

You may use either the SetAttributeAll or the Module Reconfigure message.

Set Attribute All message:

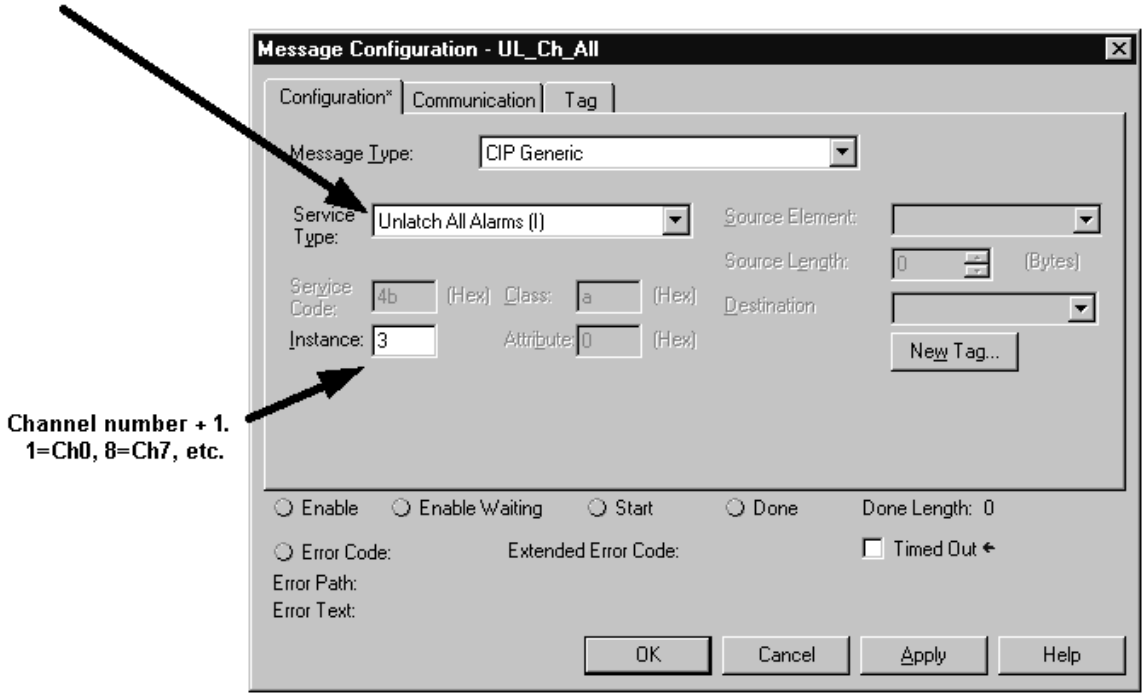


Module Reconfigure Message:



Rung 4: This rung describes how to unlatch process alarms.

Service Type	Service Code	Class	Attribute
Unlatch All Alarms(I)	4b	a	0
Unlatch Analog High Alarm (I)	4b	a	6c
Unlatch Analog High High Alarm (I)	4b	a	6e
Unlatch Analog Low Alarm (I)	4b	a	6b
Unlatch Analog Low Low Alarm (I)	4b	a	6d
Unlatch Rate Alarm (I)	4b	a	6f



Demultiplexing HART Data

The following ladder demonstrates how to demultiplex the hart data contained in the HartData tag within the input image.

Figure 8.2 (IF8H Demultiplexing Ladder)

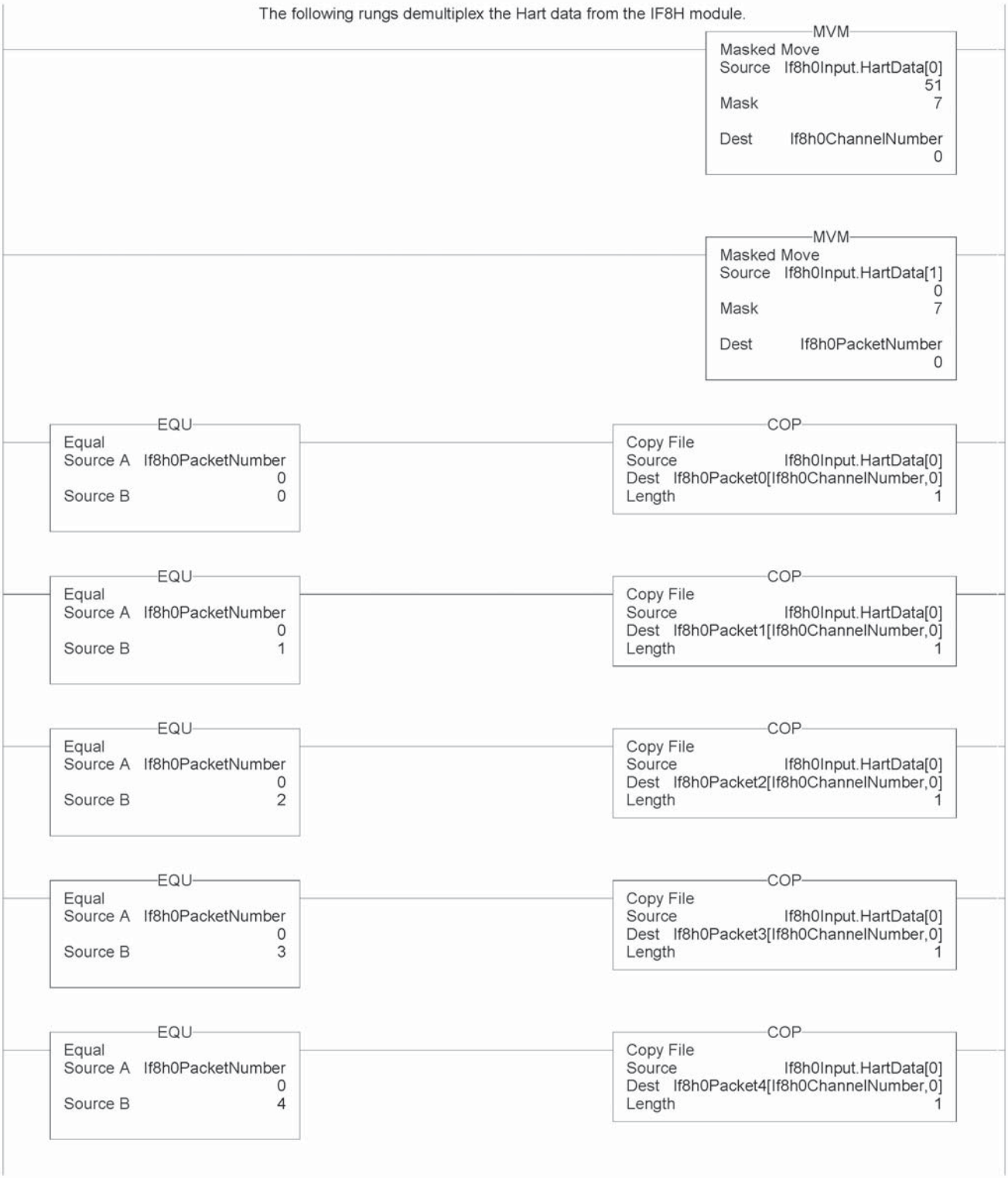
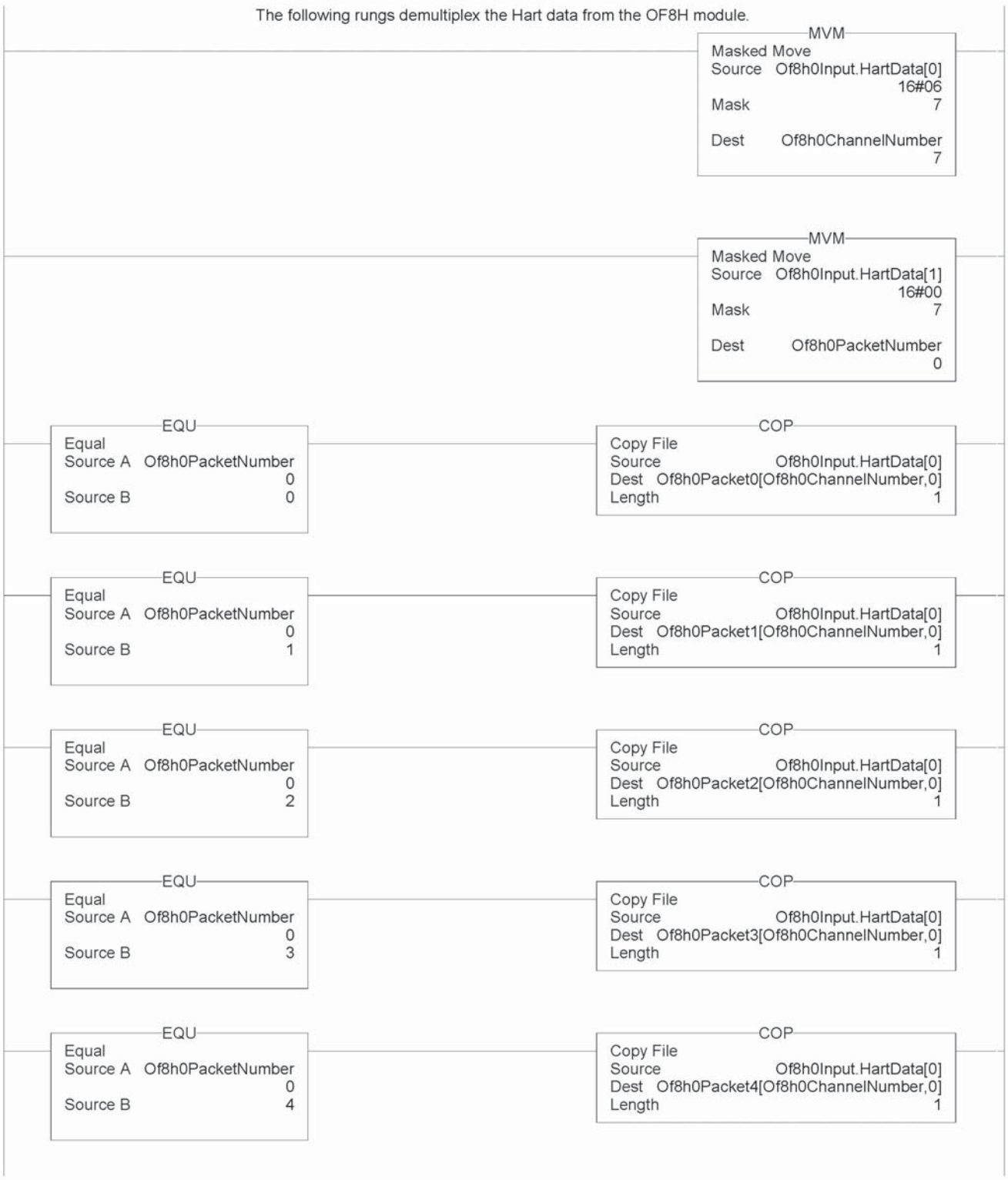


Figure 8.3 (OF8H Demultiplexing Ladder)



Sending HART Commands Using the MSG Instruction

The following ladder samples demonstrate how to send HART messages using ladder.

Figure 8.3a (IF8H HART Message Ladder)

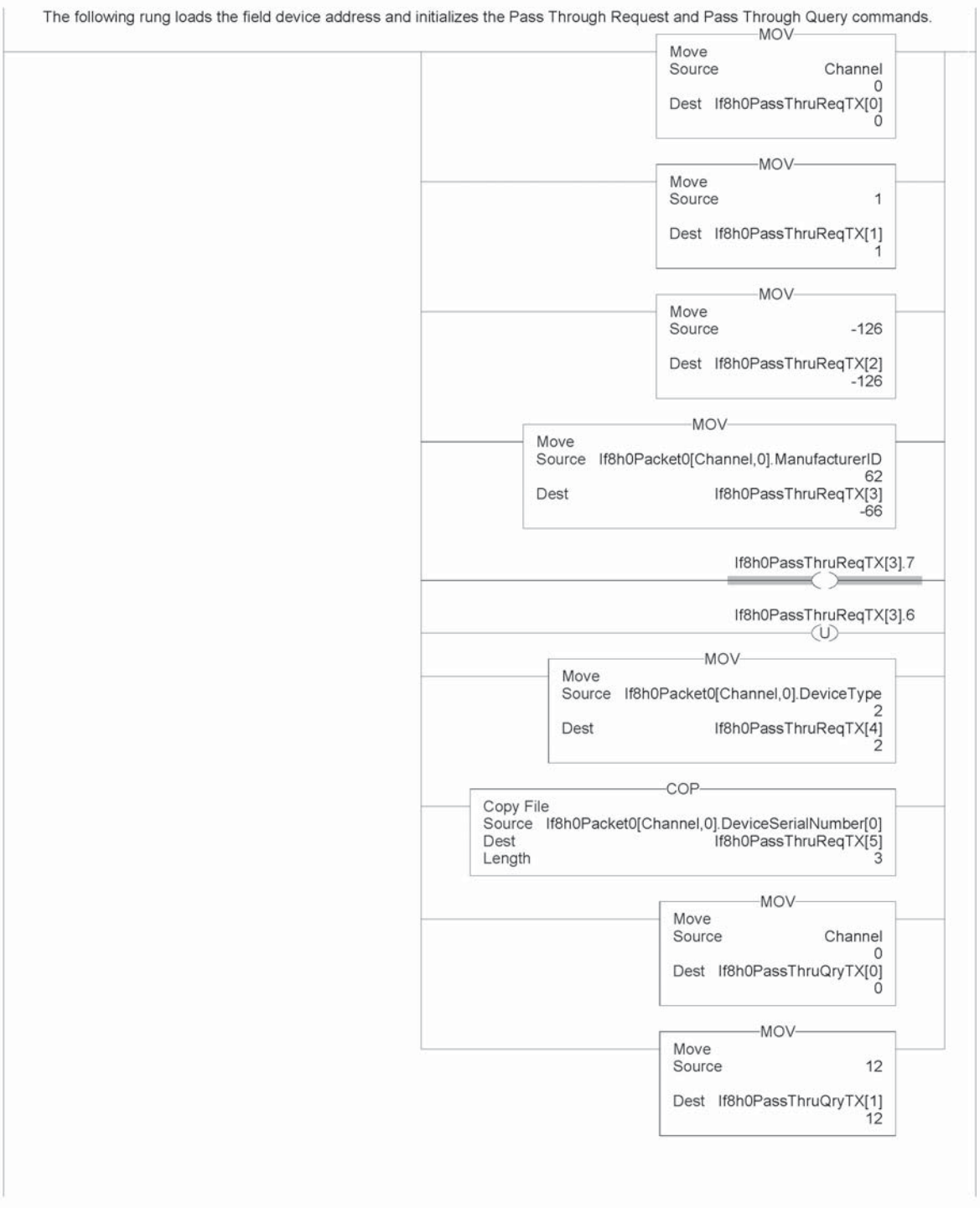


Figure 8.3b (IF8H HART Message Ladder)

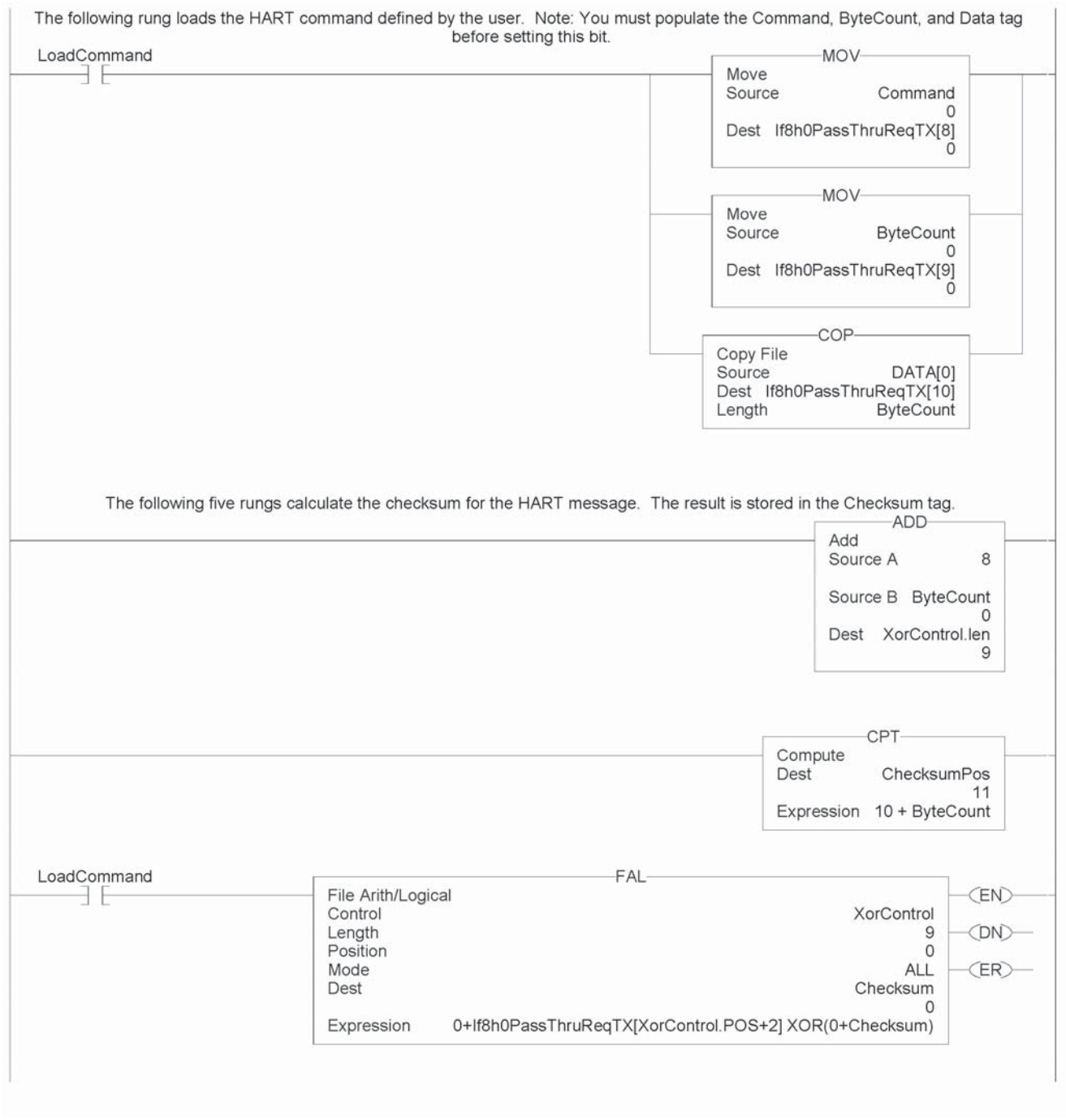


Figure 8.3c (IF8H HART Message Ladder)

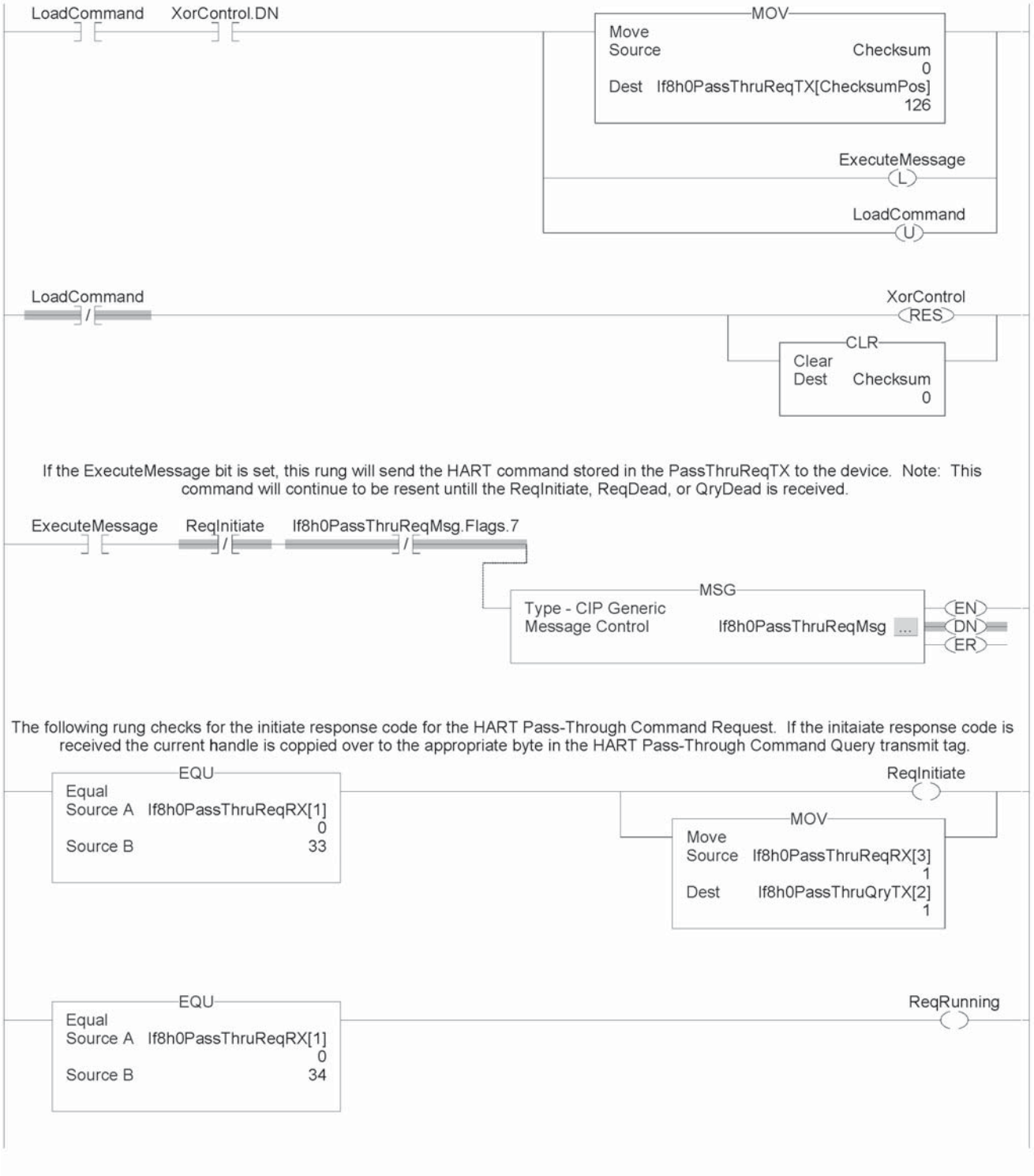


Figure 8.3d (IF8H HART Message Ladder)

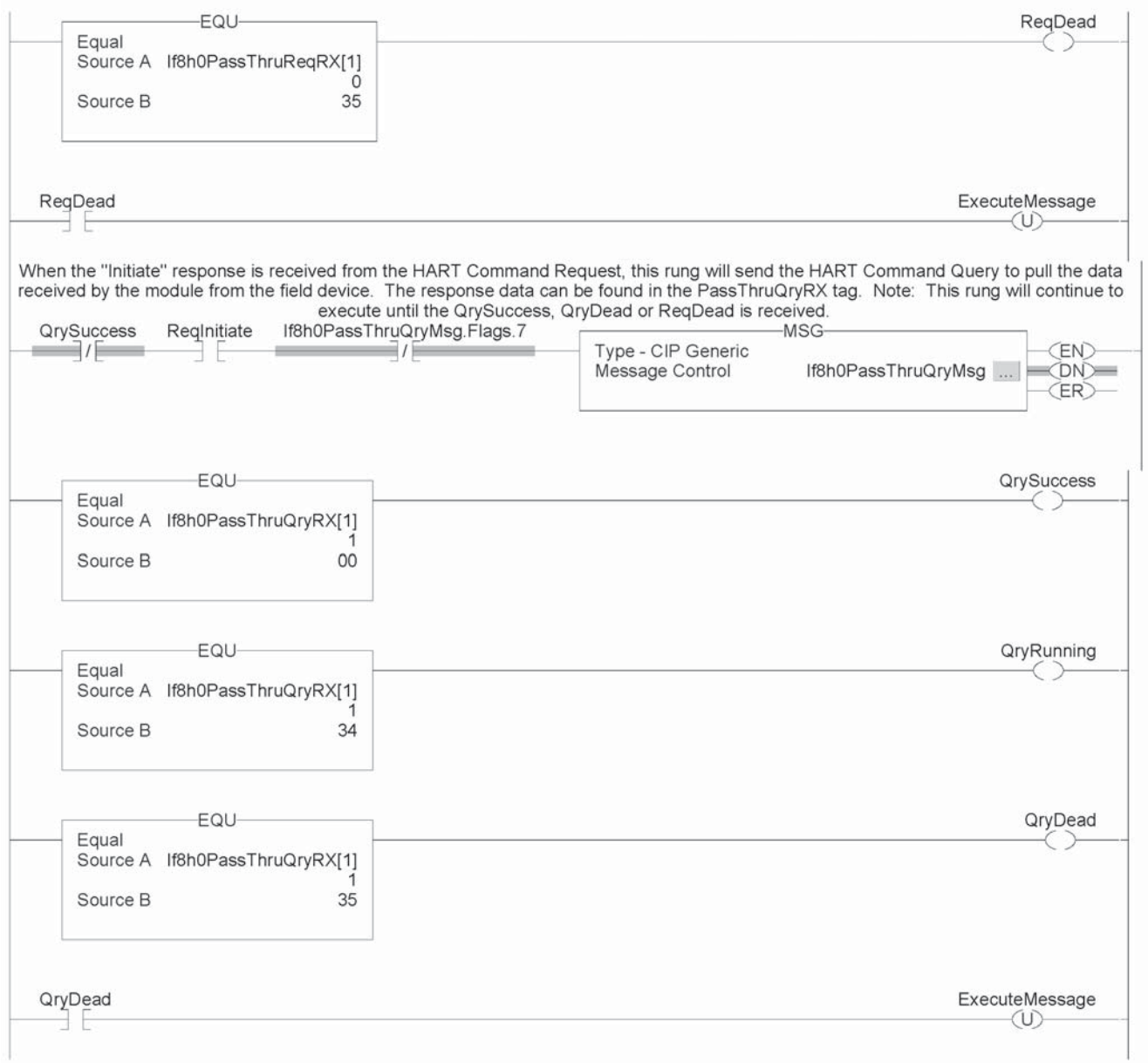


Figure 8.3e (IF8H HART Message Ladder)

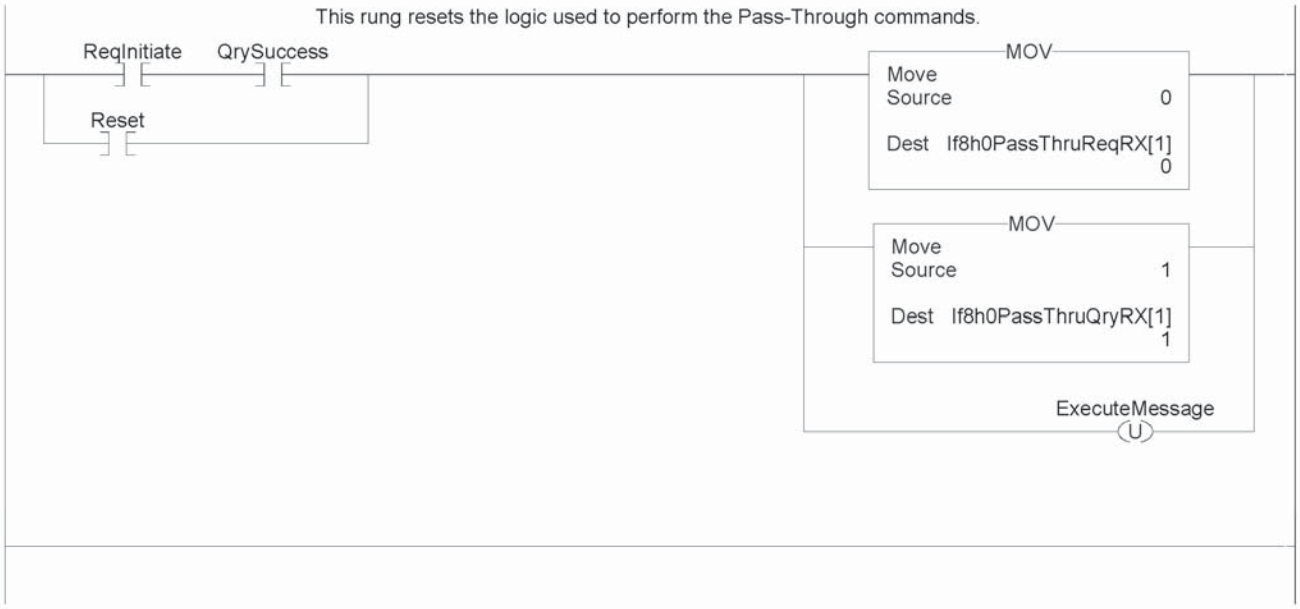


Figure 8.4a (OF8H HART Message Ladder)

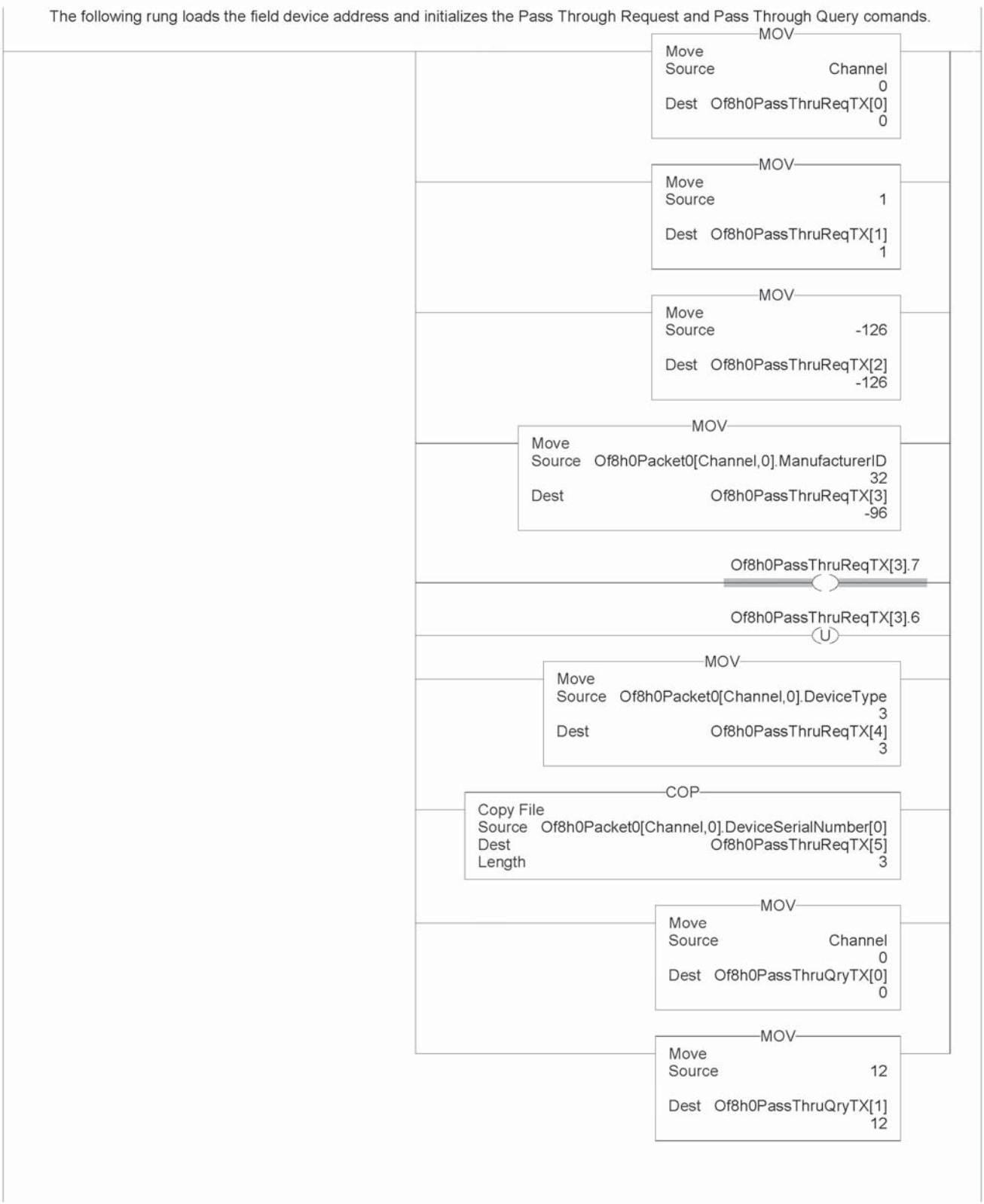


Figure 8.4b (OF8H HART Message Ladder)

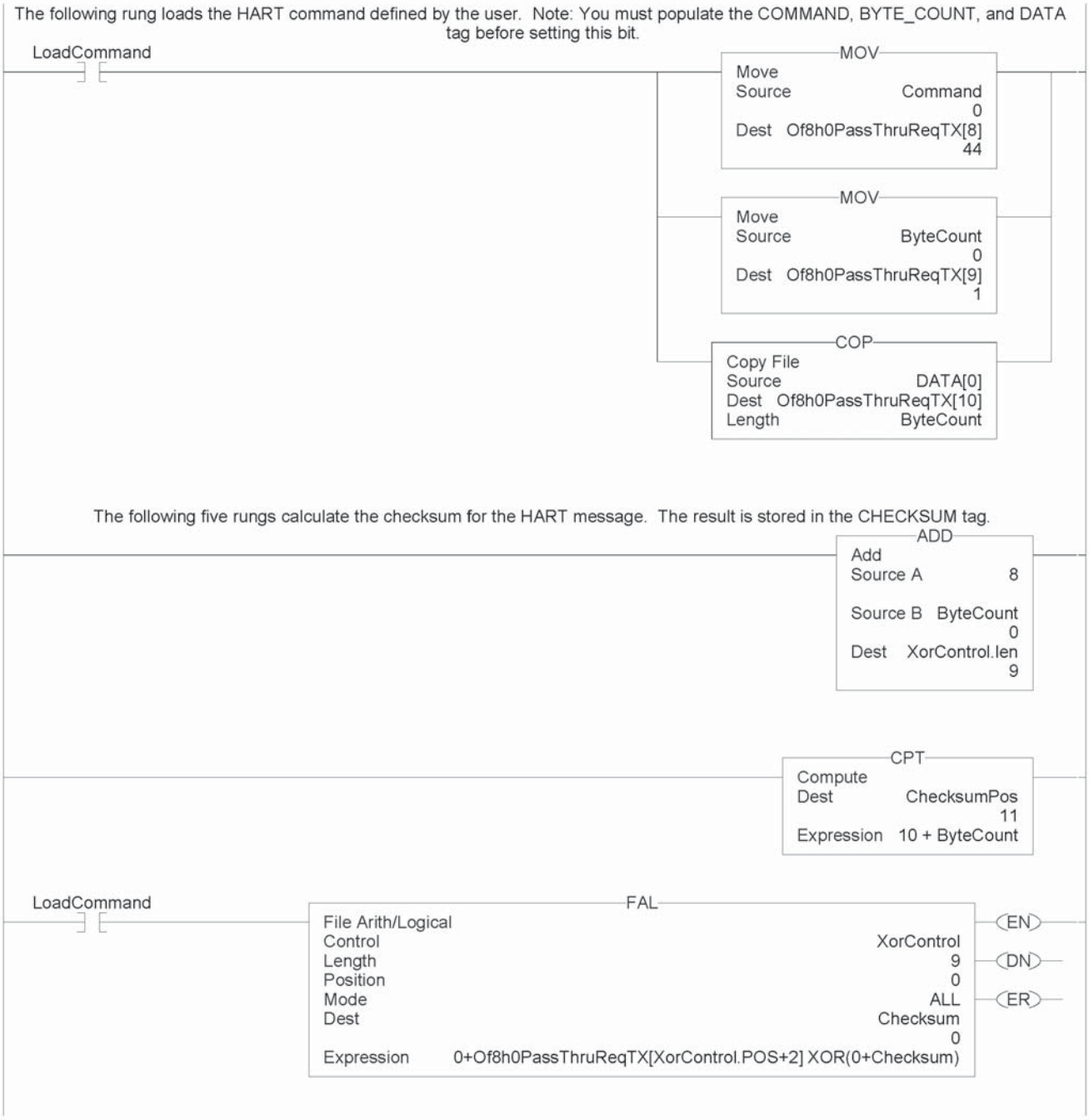


Figure 8.4c (OF8H HART Message Ladder)

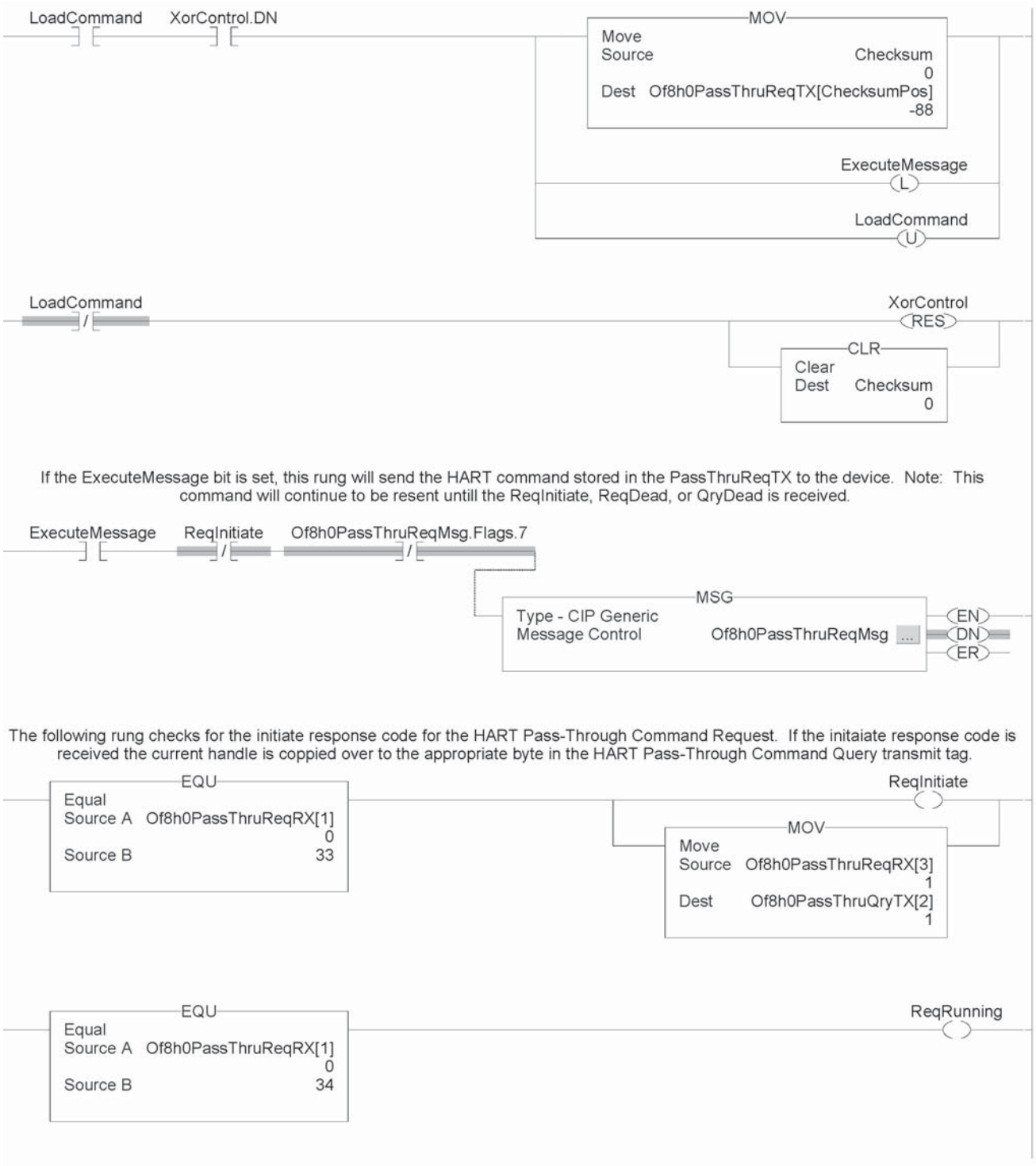


Figure 8.4d (OF8H HART Message Ladder)

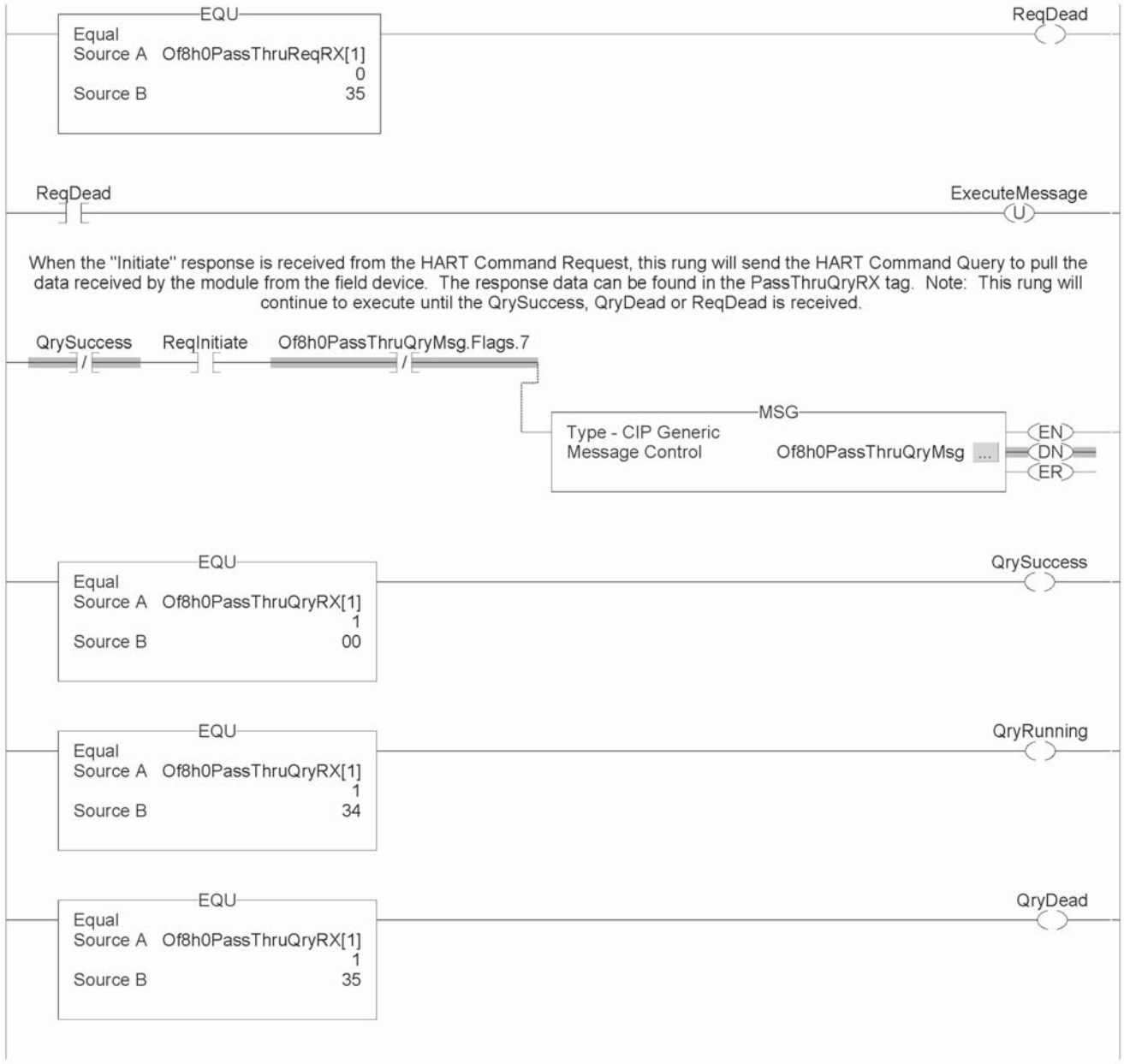
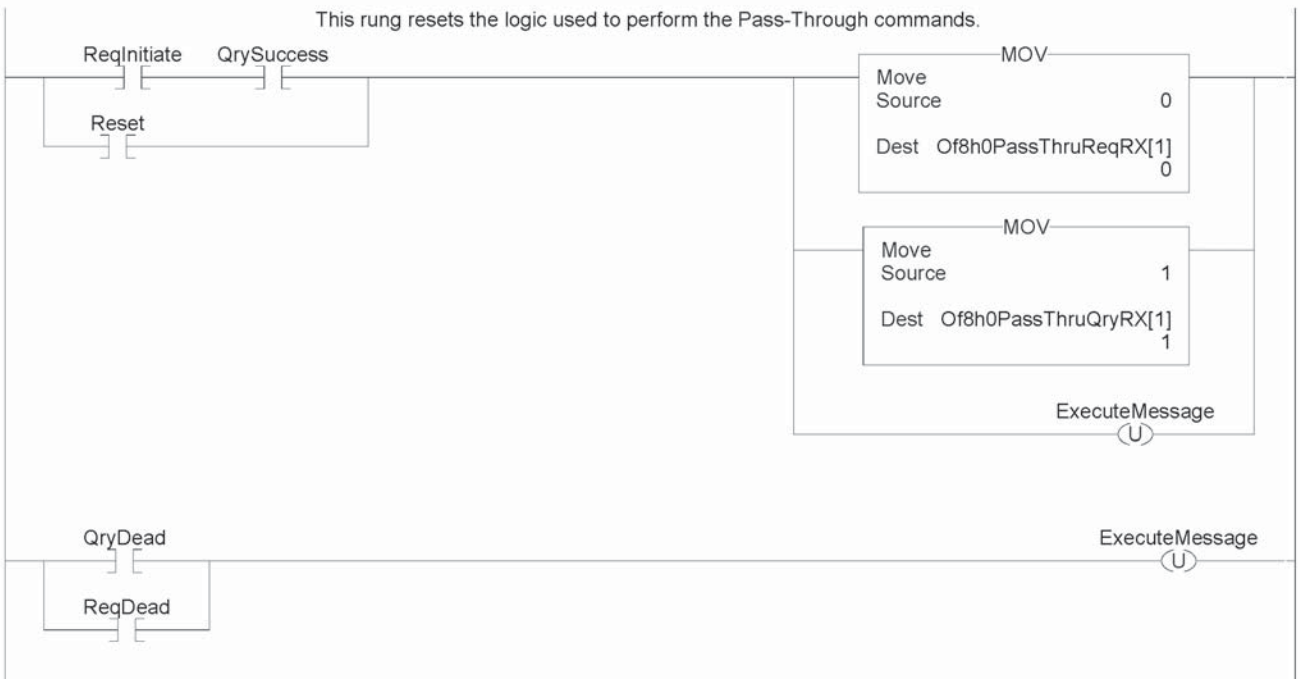


Figure 8.4e (OF8H HART Message Ladder)

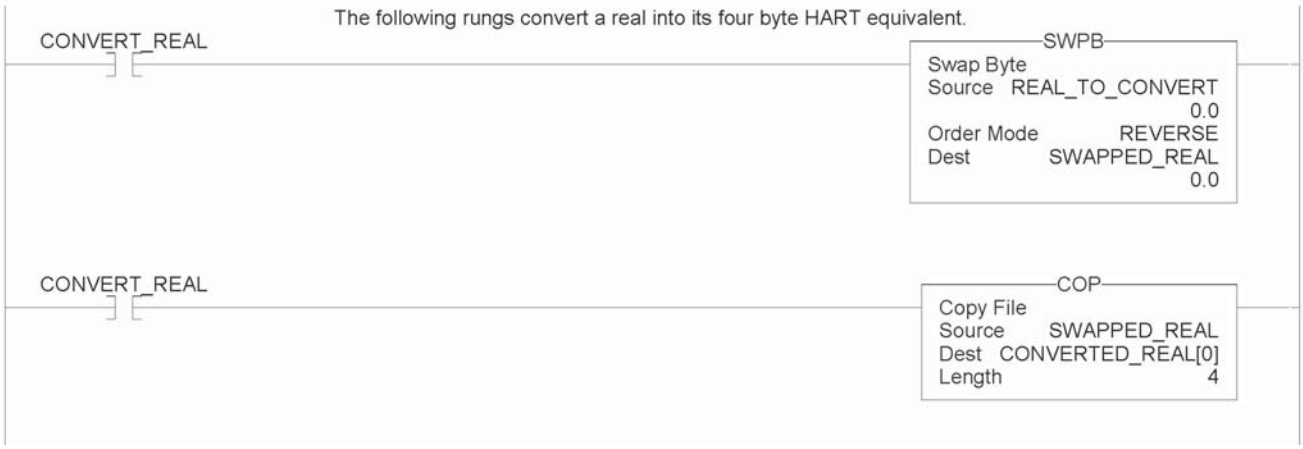


Swap Byte Ladder

Figure 8.5 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.

Attention: 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 8.5 (Converting a FLOAT Value To Its 4 byte HART Equivalent)



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: 0x82, 0x08 and 0x20.

	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 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.

Figure 5.6 demonstrates how to pack 4 unpacked ASCII characters into 3 bytes.

Figure 5.6a (Packed ASCII)

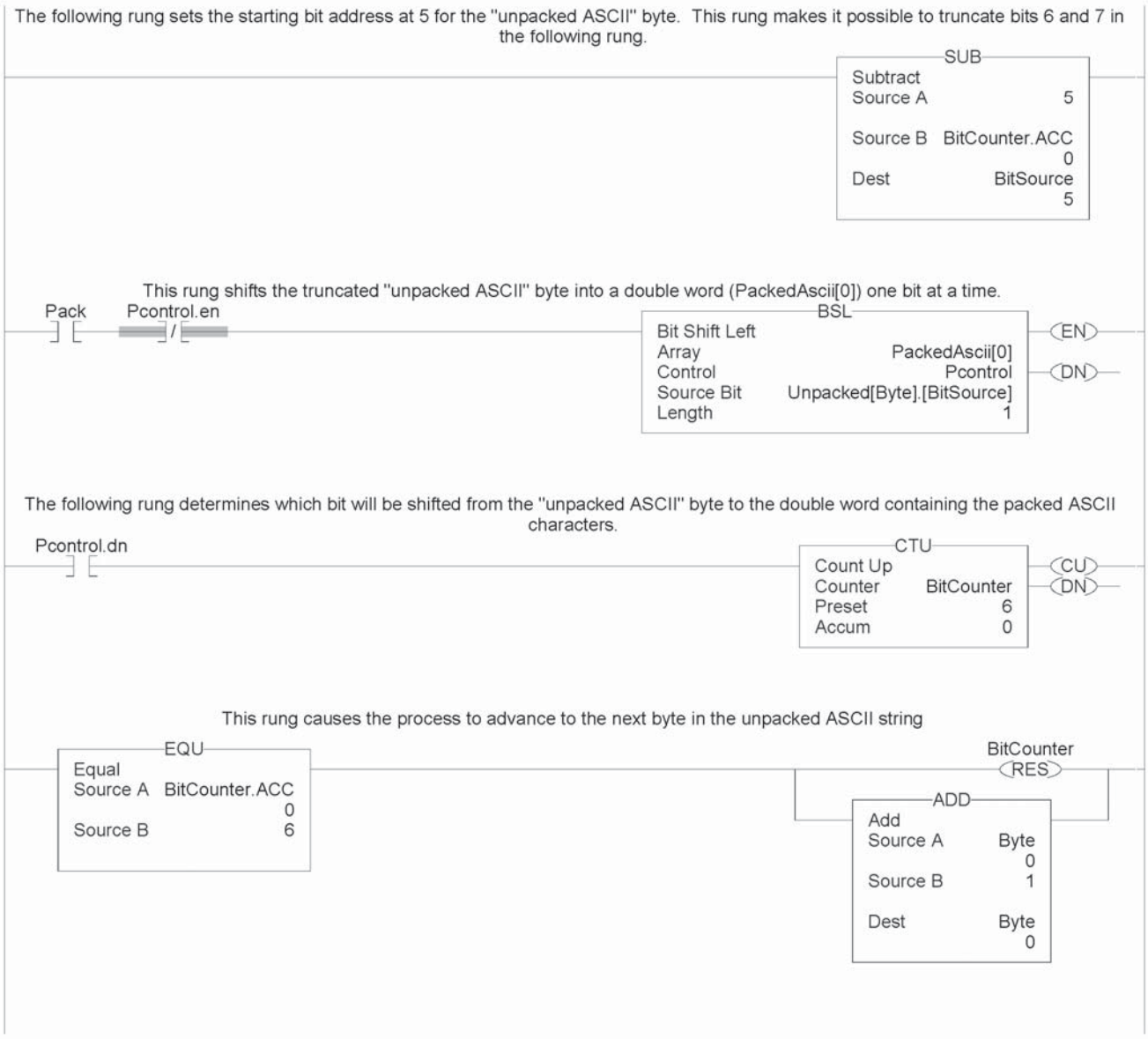
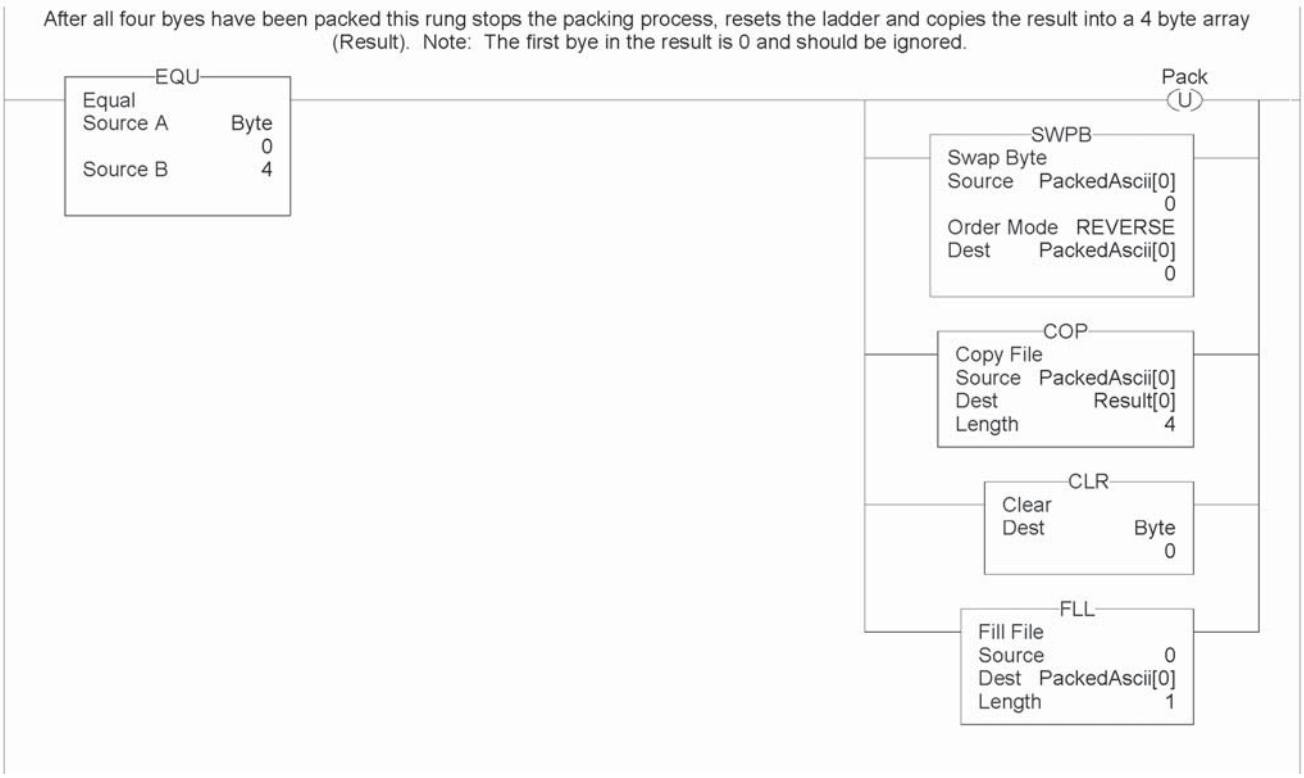


Figure 5.6b (Packed ASCII)



Troubleshooting

Using Module Indicators to Troubleshoot

The analog I/O modules have indicators which provide indication of module status. ControlLogix modules use the following:

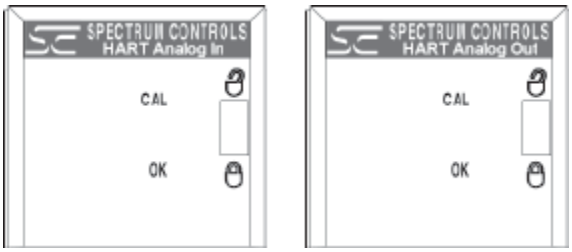
LED	This display:	Means	Take this action:
OK	Steady Green Light	The inputs are being multicast	None
OK	Flashing Green Light	The module has passed internal diagnostics but is not currently performing connected communication	None
OK	Flashing Red Light	Previously established communication has timed out and chassis communications	Check controller
OK	Steady Red Light	It is likely the module should be replaced	See below
CAL	Flashing Green Light	The module is in calibration mode	None

Under fault conditions the module will communicate a particular error via a LED blink code. A description of the fault conditions and LED blink codes is listed below...

OK LED	CAL LED	Fault Status
RED	Flashing Green	Firmware Download in Process
RED	3 Blinks	Major Nonrecoverable
	Boot code section has failed the CRC check. Send in Module for Repair	
RED	4 Blinks	Major Nonrecoverable
	Serial Number not programmed. Send in Module for Repair	
RED	5 Blinks	Major Nonrecoverable
	Boot code section has failed the CRC check. Send in Module for Repair	
RED	6 Blinks	Major Recoverable
	Application code section has failed the CRC check. Try reprogramming the module firmware. If condition persists send module in for repair.	
RED	9 Blinks	Major Nonrecoverable
	Module has lost it's calibration data. Send in Module for repair.	
RED	10 Blinks	Major Recoverable
	Module's firmware watchdog timer has timed out. Try resetting module. If condition persists send module in for repair.	

Note: In RSLogix5000 the Fault Status can be seen in the "Module Info" tab of the module's properties dialog.

The following LED displays are used with ControlLogix analog HART modules:



Using RSLogix 5000 to Troubleshoot Your Module

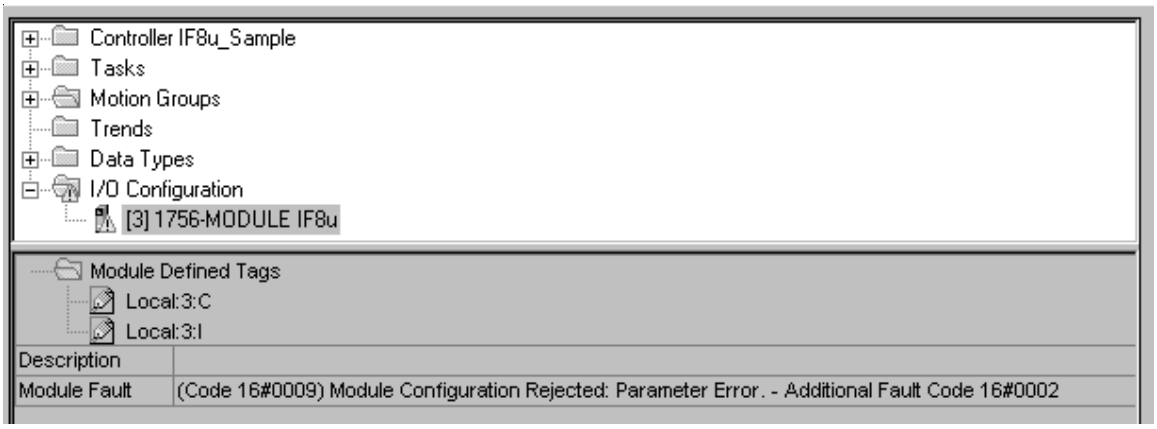
In addition to the LED display on the module, RSLogix 5000 will alert you to fault conditions. You will be alerted in one of three ways:

· Warning signal on the main screen next to the module-This occurs when the connection to the module is broken

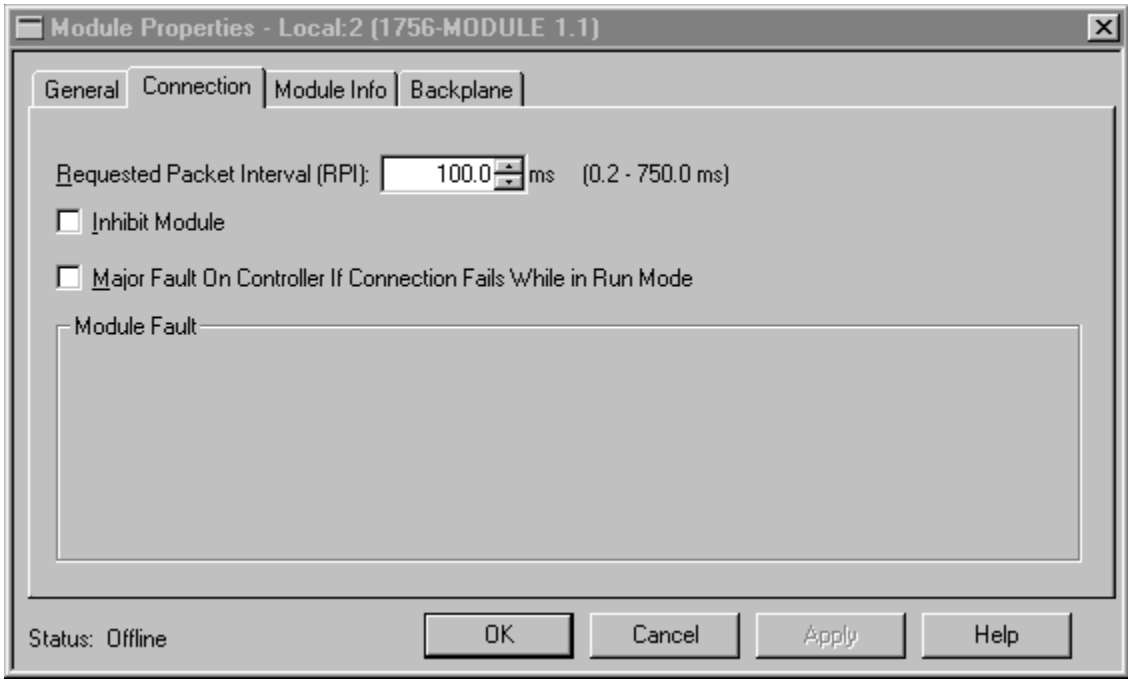
· Fault message in a screen's status line · Notification in the Tag Editor - General module faults are also reported in the Tag Editor. Diagnostic faults are only reported in the Tag Editor

· Status on the Module Info Page

The screens below display fault notification in RSLogix 5000.



Fault information on the properties screen.



Determining Fault Type

When you are monitoring a module's properties dialog in RSLogix 5000 and receive a fault message, the module fault area lists the type of fault.

Module Configuration Errors

The "Additional Fault Code" value details the configuration error if the "(16#0009) module configuration rejected: Parameter Error" was received.

1756sc-IF8H Error Codes

Extended Fault Code	Channel #	Description
0x0002	NA	INVALID FILTER
0x0003	NA	INVALID RTS
0x0004	NA	INVALID HART HANDLE TIME
0x0005	0	PROCESS ALARM LATCH NOT DISABLED
0x0006	0	RATE ALARM LATCH NOT DISABLED
0x0007	0	INVALID INPUT RANGE
0x0008	0	DIGITAL FILTER GREATER THAN TWICE THE RTS RATE
0x0009	0	INVALID RATE ALARM
0x000A	0	SIGNAL OUT OF RANGE
0x000B	0	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x000C	0	CAL BIAS SET TO NAN
0x000D	0	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x000E	0	INVALID HART RATE
0x000F	0	INVALID HIGH LOW ALARM
0x0010	0	INVALID LOW LOW ALARM
0x0011	0	INVALID HIGH HIGH ALARM
0x0012	0	INVALID ALARM DB
0x0015	1	PROCESS ALARM LATCH NOT DISABLED
0x0016	1	RATE ALARM LATCH NOT DISABLED
0x0017	1	INVALID INPUT RANGE
0x0018	1	DIGITAL FILTER GREATER THAN TWICE THE RTS RATE
0x0019	1	INVALID RATE ALARM
0x001A	1	SIGNAL OUT OF RANGE
0x001B	1	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x001C	1	CAL BIAS SET TO NAN
0x001D	1	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x001E	1	INVALID HART RATE
0x001F	1	INVALID HIGH LOW ALARM
0x0020	1	INVALID LOW LOW ALARM
0x0021	1	INVALID HIGH HIGH ALARM
0x0022	1	INVALID ALARM DB
0x0025	2	PROCESS ALARM LATCH NOT DISABLED
0x0026	2	RATE ALARM LATCH NOT DISABLED
0x0027	2	INVALID INPUT RANGE
0x0028	2	DIGITAL FILTER GREATER THAN TWICE THE RTS RATE
0x0029	2	INVALID RATE ALARM
0x002A	2	SIGNAL OUT OF RANGE
0x002B	2	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL

Extended Fault Code	Channel #	Description
0x002C	2	CAL BIAS SET TO NAN
0x002D	2	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x002E	2	INVALID HART RATE
0x002F	2	INVALID HIGH LOW ALARM
0x0030	2	INVALID LOW LOW ALARM
0x0031	2	INVALID HIGH HIGH ALARM
0x0032	2	INVALID ALARM DB
0x0035	3	PROCESS ALARM LATCH NOT DISABLED
0x0036	3	RATE ALARM LATCH NOT DISABLED
0x0037	3	INVALID INPUT RANGE
0x0038	3	DIGITAL FILTER GREATER THAN TWICE THE RTS RATE
0x0039	3	INVALID RATE ALARM
0x003A	3	SIGNAL OUT OF RANGE
0x003B	3	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x003C	3	CAL BIAS SET TO NAN
0x003D	3	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x003E	3	INVALID HART RATE
0x003F	3	INVALID HIGH LOW ALARM
0x0040	3	INVALID LOW LOW ALARM
0x0041	3	INVALID HIGH HIGH ALARM
0x0042	3	INVALID ALARM DB
0x0045	4	PROCESS ALARM LATCH NOT DISABLED
0x0046	4	RATE ALARM LATCH NOT DISABLED
0x0047	4	INVALID INPUT RANGE
0x0048	4	DIGITAL FILTER GREATER THAN TWICE THE RTS RATE
0x0049	4	INVALID RATE ALARM
0x004A	4	SIGNAL OUT OF RANGE
0x004B	4	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x004C	4	CAL BIAS SET TO NAN
0x004D	4	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x004E	4	INVALID HART RATE
0x004F	4	INVALID HIGH LOW ALARM
0x0050	4	INVALID LOW LOW ALARM
0x0051	4	INVALID HIGH HIGH ALARM
0x0052	4	INVALID ALARM DB
0x0055	5	PROCESS ALARM LATCH NOT DISABLED
0x0056	5	RATE ALARM LATCH NOT DISABLED
0x0057	5	INVALID INPUT RANGE

Extended Fault Code	Channel #	Description
0x0058	5	DIGITAL FILTER GREATER THAN TWICE THE RTS RATE
0x0059	5	INVALID RATE ALARM
0x005A	5	SIGNAL OUT OF RANGE
0x005B	5	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x005C	5	CAL BIAS SET TO NAN
0x005D	5	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x005E	5	INVALID HART RATE
0x005F	5	INVALID HIGH LOW ALARM
0x0060	5	INVALID LOW LOW ALARM
0x0061	5	INVALID HIGH HIGH ALARM
0x0062	5	INVALID ALARM DB
0x0065	6	PROCESS ALARM LATCH NOT DISABLED
0x0066	6	RATE ALARM LATCH NOT DISABLED
0x0067	6	INVALID INPUT RANGE
0x0068	6	DIGITAL FILTER GREATER THAN TWICE THE RTS RATE
0x0069	6	INVALID RATE ALARM
0x006A	6	SIGNAL OUT OF RANGE
0x006B	6	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x006C	6	CAL BIAS SET TO NAN
0x006D	6	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x006E	6	INVALID HART RATE
0x006F	6	INVALID HIGH LOW ALARM
0x0070	6	INVALID LOW LOW ALARM
0x0071	6	INVALID HIGH HIGH ALARM
0x0072	6	INVALID ALARM DB
0x0075	7	PROCESS ALARM LATCH NOT DISABLED
0x0076	7	RATE ALARM LATCH NOT DISABLED
0x0077	7	INVALID INPUT RANGE
0x0078	7	DIGITAL FILTER GREATER THAN TWICE THE RTS RATE
0x0079	7	INVALID RATE ALARM
0x007A	7	SIGNAL OUT OF RANGE
0x007B	7	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x007C	7	CAL BIAS SET TO NAN
0x007D	7	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x007E	7	INVALID HART RATE
0x007F	7	INVALID HIGH LOW ALARM
0x0080	7	INVALID LOW LOW ALARM
0x0081	7	INVALID HIGH HIGH ALARM
0x0082	7	INVALID ALARM DB

1756sc-OF8H Error Codes

Extended Fault Code	Channel #	Description
0x0001	NA	INVALID REVISION NUMBER
0x0004	NA	INVALID HART HANDLE TIME
0x0005	0	BAD RAMP LATCH
0x0006	0	BAD CLAMP LATCH
0x000A	0	BAD RAMP TO IDLE
0x000B	0	BAD RAMP TO FAULT
0x000C	0	INVALID INPUT RANGE
0x000D	0	BAD MAX RAMP
0x000E	0	BAD FAULT VALUE
0x000F	0	BAD IDLE VALUE
0x0010	0	SIGNAL OUT OF RANGE
0x0011	0	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x0012	0	CAL BIAS SET TO NAN
0x0013	0	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x0014	0	INVALID HART RATE
0x0015	0	BAD CLAMP
0x001B	1	BAD RAMP LATCH
0x001C	1	BAD CLAMP LATCH
0x0020	1	BAD RAMP TO IDLE
0x0021	1	BAD RAMP TO FAULT
0x0022	1	INVALID INPUT RANGE
0x0023	1	BAD MAX RAMP
0x0024	1	BAD FAULT VALUE
0x0025	1	BAD IDLE VALUE
0x0026	1	SIGNAL OUT OF RANGE
0x0027	1	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x0028	1	CAL BIAS SET TO NAN
0x0029	1	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x002A	1	INVALID HART RATE
0x002B	1	BAD CLAMP
0x0031	2	BAD RAMP LATCH
0x0032	2	BAD CLAMP LATCH
0x0036	2	BAD RAMP TO IDLE
0x0037	2	BAD RAMP TO FAULT
0x0038	2	INVALID INPUT RANGE
0x0039	2	BAD MAX RAMP

Extended Fault Code	Channel #	Description
0x003A	2	BAD FAULT VALUE
0x003B	2	BAD IDLE VALUE
0x003C	2	SIGNAL OUT OF RANGE
0x003D	2	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x003E	2	CAL BIAS SET TO NAN
0x003F	2	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x0040	2	INVALID HART RATE
0x0041	2	BAD CLAMP
0x0047	3	BAD RAMP LATCH
0x0048	3	BAD CLAMP LATCH
0x004C	3	BAD RAMP TO IDLE
0x004D	3	BAD RAMP TO FAULT
0x004E	3	INVALID INPUT RANGE
0x004F	3	BAD MAX RAMP
0x0050	3	BAD FAULT VALUE
0x0051	3	BAD IDLE VALUE
0x0052	3	SIGNAL OUT OF RANGE
0x0053	3	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x0054	3	CAL BIAS SET TO NAN
0x0055	3	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x0056	3	INVALID HART RATE
0x0057	3	BAD CLAMP
0x005D	4	BAD RAMP LATCH
0x005E	4	BAD CLAMP LATCH
0x0062	4	BAD RAMP TO IDLE
0x0063	4	BAD RAMP TO FAULT
0x0064	4	INVALID INPUT RANGE
0x0065	4	BAD MAX RAMP
0x0066	4	BAD FAULT VALUE
0x0067	4	BAD IDLE VALUE
0x0068	4	SIGNAL OUT OF RANGE
0x0069	4	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x006A	4	CAL BIAS SET TO NAN
0x006B	4	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x006C	4	INVALID HART RATE
0x006D	4	BAD CLAMP
0x0073	5	BAD RAMP LATCH

Extended Fault Code	Channel #	Description
0x0074	5	BAD CLAMP LATCH
0x0078	5	BAD RAMP TO IDLE
0x0079	5	BAD RAMP TO FAULT
0x007A	5	INVALID INPUT RANGE
0x007B	5	BAD MAX RAMP
0x007C	5	BAD FAULT VALUE
0x007D	5	BAD IDLE VALUE
0x007E	5	SIGNAL OUT OF RANGE
0x007F	5	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x0080	5	CAL BIAS SET TO NAN
0x0081	5	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x0082	5	INVALID HART RATE
0x0083	5	BAD CLAMP
0x0089	6	BAD RAMP LATCH
0x008A	6	BAD CLAMP LATCH
0x008E	6	BAD RAMP TO IDLE
0x008F	6	BAD RAMP TO FAULT
0x0090	6	INVALID INPUT RANGE
0x0091	6	BAD MAX RAMP
0x0092	6	BAD FAULT VALUE
0x0093	6	BAD IDLE VALUE
0x0094	6	SIGNAL OUT OF RANGE
0x0095	6	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x0096	6	CAL BIAS SET TO NAN
0x0097	6	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x0098	6	INVALID HART RATE
0x0099	6	BAD CLAMP
0x009F	7	BAD RAMP LATCH
0x00A0	7	BAD CLAMP LATCH
0x00A4	7	BAD RAMP TO IDLE
0x00A5	7	BAD RAMP TO FAULT
0x00A6	7	INVALID INPUT RANGE
0x00A7	7	BAD MAX RAMP
0x00A8	7	BAD FAULT VALUE
0x00A9	7	BAD IDLE VALUE
0x00AA	7	SIGNAL OUT OF RANGE
0x00AB	7	LOW SIGNAL GREATER THAN OR EQUAL TO HIGH SIGNAL
0x00AC	7	CAL BIAS SET TO NAN
0x00AD	7	HIGH ENGINEERING EQUAL TO LOW ENGINEERING
0x00AE	7	INVALID HART RATE
0x00AF	7	BAD CLAMP

Maintaining Your Module And Ensuring Safety

Read this chapter to familiarize yourself with:

- preventive maintenance
- safety considerations

The National Fire Protection Association (NFPA) recommends maintenance procedures for electrical equipment. Refer to article 70B of the NFPA for general safety-related work practices.

Preventive Maintenance

The printed circuit boards of your module must be protected from dirt, oil, moisture, and other airborne contaminants. To protect these boards, install the ControlLogix system in an enclosure suitable for its operating environment. Keep the interior of the enclosure clean, and whenever possible, keep the enclosure door closed.

Also, regularly inspect the terminal connections for tightness. Loose connections may cause a malfunctioning of the system or damage to the components.

WARNING

POSSIBLE LOOSE CONNECTIONS

Before inspecting connections, always ensure that incoming power is OFF.

Failure to observe this precaution can cause personal injury and equipment damage.

Safety Considerations

Safety is always the most important consideration. Actively think about the safety of yourself and others, as well as the condition of your equipment. The following are some things to consider:

Indicator Lights – When the module status LED on your module is illuminated, your module is receiving power.

Activating Devices When Troubleshooting – Never reach into a machine to activate a device; the machine may move unexpectedly. Use a wooden stick.

Standing Clear Of Machinery – When troubleshooting a problem with any ControlLogix system, have all personnel remain clear of machinery. The problem may be intermittent, and the machine may move unexpectedly. Have someone ready to operate an emergency stop switch.

CAUTION

POSSIBLE EQUIPMENT OPERATION

Never reach into a machine to actuate a switch. Also, remove all electrical power at the main power disconnect switches before checking electrical connections or inputs/outputs causing machine motion.

Failure to observe these precautions can cause personal injury or equipment damage.

Safety Circuits – Circuits installed on machinery for safety reasons (like over-travel limit switches, stop push-buttons, and interlocks) should always be hard-wired to the master control relay. These circuits should also be wired in series so that when any one circuit opens, the master control relay is de-energized, thereby removing power. Never modify these circuits to defeat their function. Serious injury or equipment damage may result.

WARNING

EXPLOSION HAZARD

SUBSTITUTION OF COMPONENTS MAY IMPAIR SUITABILITY FOR CLASSI DIVISION2.

WARNING

EXPLOSION HAZARD

DO NOT DISCONNECT EQUIPMENT UNLESS POWER HAS BEEN SWITCHED OFF OR THE AREA IS KNOWN TO BE NON-HAZARDOUS

NOTE: THIS EQUIPMENT IS SUITABLE FOR USE IN CLASSI, DIVISION 2, GROUPS A, B, C, AND D OR NON-HAZARDOUS LOCATIONS ONLY.

WARNING

EXPLOSION HAZARD

WHEN IN HAZARDOUS LOCATIONS, TURN OFF POWER BEFORE REPLACING OR WIRING MODULES.

WARNING

THIS DEVICE IS INTENDED TO ONLY BE USED WITH THE ALLEN-BRADLEY CONTROLLOGIX 1756 I/O SYSTEM.

WARNING

ELECTROSTATICALLY SENSITIVE COMPONENTS
EXPLOSION HAZARD
Substitution of components may impair suitability for Class I, Division 2.

1756sc-IF8H Module Specifications

This appendix lists the specifications for the 1756sc-IF8H Analog Input module.

Electrical Specifications 1756sc-IF8H

Parameter	Specification
Number of Inputs	8 single ended voltage or current inputs
Module Location	ControlLogix or ProcessLogix Chassis
Power Consumption	300mA @ 5.1V 70mA @ 24V
Power Dissipation within Module	3.21W – voltage / 4.01W – current
Thermal Dissipation	11.0 BTU/hr – voltage / 13.7 BTU/hr – current
Input Ranges	+/-10.25 V 0 – 10.25 V 0 – 5.125 V 0 – 20.58 mA 3.42 – 20.58 mA
Data Format	IEEE 754 32 bit Floating point
Input Impedance	> 1 MΛ – voltage 249 ohms – current
Open Circuit Detection Time	5 seconds
Input Overvoltage Protection	30V DC – voltage 8V DC – current

Normal Mode Noise Rejection

Filter Frequency	-3dB Frequency	50 Hz Rejection (dB)	60 Hz Rejection (dB)	Effective Resolution
10 Hz	7.80 Hz	95	97	16
15 Hz	11.70 Hz	85	88	16
20 Hz	15.60 Hz	38	65	16
50 Hz	39.30 Hz	4	7	16
60 Hz	39.30 Hz	4	7	16
100 Hz	65.54 Hz	2	2.5	16
250 Hz	163.9 Hz	0.5	0.6	14
1000 Hz	659.7 Hz	0.1	0.1	12

Common Mode Noise Rejection	> 100 dB at 50/60Hz (10 Hz filter)
Calibrated Accuracy at 25°C	Better than 0.05% of range – voltage Better than 0.15% of range – current
Input Offset Drift with Temperature	<= 90 uV/°C

Gain Drift with Temperature	10 ppm/°C – voltage 20 ppm/°C – current
Module Error over Full Temp. Range	0.1% of range – voltage 0.3% of range – current
Minimum Module Scan Time for all Channels	
Analog	18 – 488 msec (filter dependent)
HART	6 – 7 seconds typical with no pass-thru or device information messaging active
Isolation Voltage	2550 VDC terminal block to backplane (1 second)
Module Conversion Method	Sigma-Delta ADC
RTB Screw Torque	4.4 inch-pounds (0.4 Nm)
Module Keying	(Backplane)
RTB Keying	User defined
Field Wiring Arm and Housing	36 Position RTB (1756-TBCH or TBS6H)
Conductors Wire Size	22-14 gauge (2mm) stranded maximum 3/64 inch (1.2mm) insulation maximum

Environmental Conditions

Operating Temperature	0 to 60°C (32 to 140°F)
Storage Temperature	-40 to 85°C (-40 to 185°F)
Relative Humidity	5 to 95% noncondensing

Regulatory Compliance

UL 508
73/23/ EEC Low Voltage Directive
89/336/ EEC Electromagnetic Compatibility
CSA (Class 1, Div 2, Group A,B,C,D)
CE compliance to EN 61010-1 and EN 61131-2, EN61000-6-2:2001, EN61000-6-4:2001
EN61010-1:2001, EN61131-2:1994 + A11:1996 + A12:2000
FM (Class 1, Div 2, Group A,B,C,D)

Firmware Revision History

Revision No.	Description
1.1	New
2.3	Updated to provide user selectable pass through queue servicing speed.

1756sc-OF8H Module Specifications

This appendix lists the specifications for the 1756sc-OF8H Analog Input HART Module.

Specifications 1756sc-OF8H

Parameter	Specification
Number of Outputs	8 voltage or current outputs
Module Location	ControlLogix or ProcessLogix chassis
Power Consumption	200mA @ 5.1 V 230mA @ 24 V
Power Dissipation within Module	
Output Range Current	0 to 21mA
Output Range Voltage	+/- 10.4V
Resolution	15 bits across 21mA - 650nA/bit 16 bits across –10.4V to +10.4V
Data Format	Floating point IEEE 32 bit
Open Circuit Detection	Current output only (Output must be set to ³0.1mA)
Output Overvoltage Protection	±24V dc
Output Short Circuit Protection	Current outputs—Electronically current limited to 21mA or less with no damage.
Voltage Outputs	Current limited to 35mA or less with no damage.
Drive Capability	50 to 750 ohm with short circuit survival. Voltage Outputs— ³2K ohm at 10.4V
Load Reactance	10uH max (Current) 1uF max (Voltage)
Output Settling Time	
Current Output (No Hart)	<23ms to 95% with resistive loads
Current Output (With Hart)	<37ms to 95% with resistive loads
Voltage Output	<8.5ms to 95% with resistive loads
Calibrated Accuracy at 25°C	Better than 0.1 % of range for voltage outputs .15% of range for current outputs
Calibration interval	12 months typical.
Module Error over Full Temp. Range	0.15% of range - voltage 0.3% of range - current 4 to 21mA
Output Offset drift with temperature.	100uV per degree C typical 200nA/degree typical
Gain drift with temperature	20ppm/degree C voltage maximum 35ppm/deg C current maximum

Module Scan Time for all Channels—	
Analog	12ms minimum floating point
HART (Typical)	6.4 seconds with 4 variables 12 seconds with 8 variables.
Isolation Voltage	2550VDC terminal block to backplane for 1 second. 2550VDC Channel to Frame Ground for 1 second.
Module Conversion Method	R-Ladder DAC
RTB Screw Torque (NEMA)	7-9 inch-pounds (0.8-1Nm)
Module Keying (Backplane)	Electronic
RTB Keying	User defined
Field Wiring Arm and Housing	20 Position RTB (1756-TBNH or TBSH)1

Environmental Conditions

Operating Temperature	0 to 60°C (32 to 140°F)
Storage Temperature	-40 to 85°C (-40 to 185°F)
Relative Humidity	5 to 95% noncondensing

Regulatory Compliance

UL 508
CSA (Class 1, Div 2, Group A,B,C,D)
CE compliance to EN 61010-1 and EN 61131-2
FM (Class 1, Div 2, Group A,B,C,D)

Firmware Revision History

Revision No.	Description
1.1	New
2.3	Updated to provide user selectable pass through queue servicing speed.

Programming Your Module

This chapter explains how program your module in the ControlLogix system. It also describes how to the module's input configuration are incorporated into your ladder logic program. Topics discussed include:

- importing the module's configuration profile
- reviewing accessing and altering configuration options.
- configuring the modules input type and filter settings
- configuring alarms and limits

Module Installation

Incorporating your module into the system is similar to adding any type of I/O module. You will use your RSLogix 5000 programming software. The module is not currently in the pick list of this software so you will use the Generic 1756 Module option as your starting point. This feature allows you to import the configuration database into your project and use ladder logic to set the attributes of each tag. These settings control features such as the modules input type, channel input range, data format, filter frequency, etc.

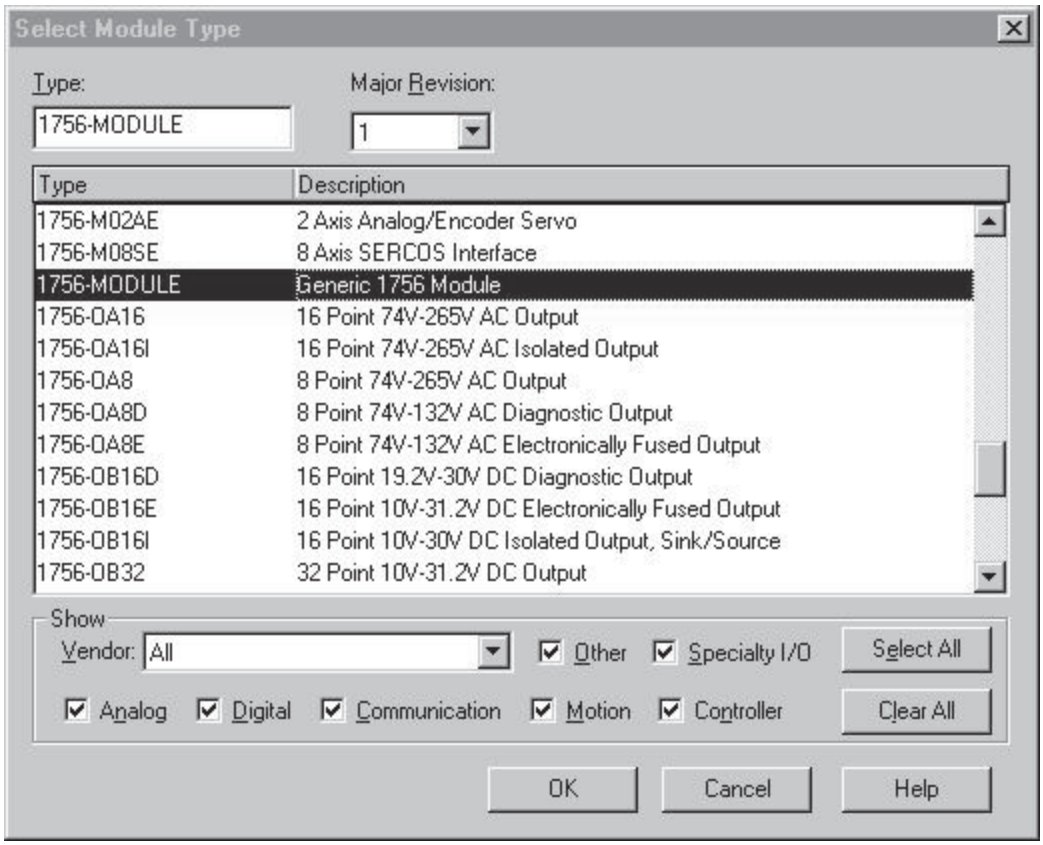
You will need to download the sample project from our website and then import this into your program. Then you may access the controller tags to configure the module. Ladder logic samples are also provided with this sample project.

Adding Your Module to a Project

The module has a unique set of tag definitions which are used to configure specific features. Chapter 5, *Channel Configuration, Data, and Status*, gives you detailed information about the data content of the configuration. These values are set using your programming software and ladder logic. Before you can use these feature you must first include the module into the project.

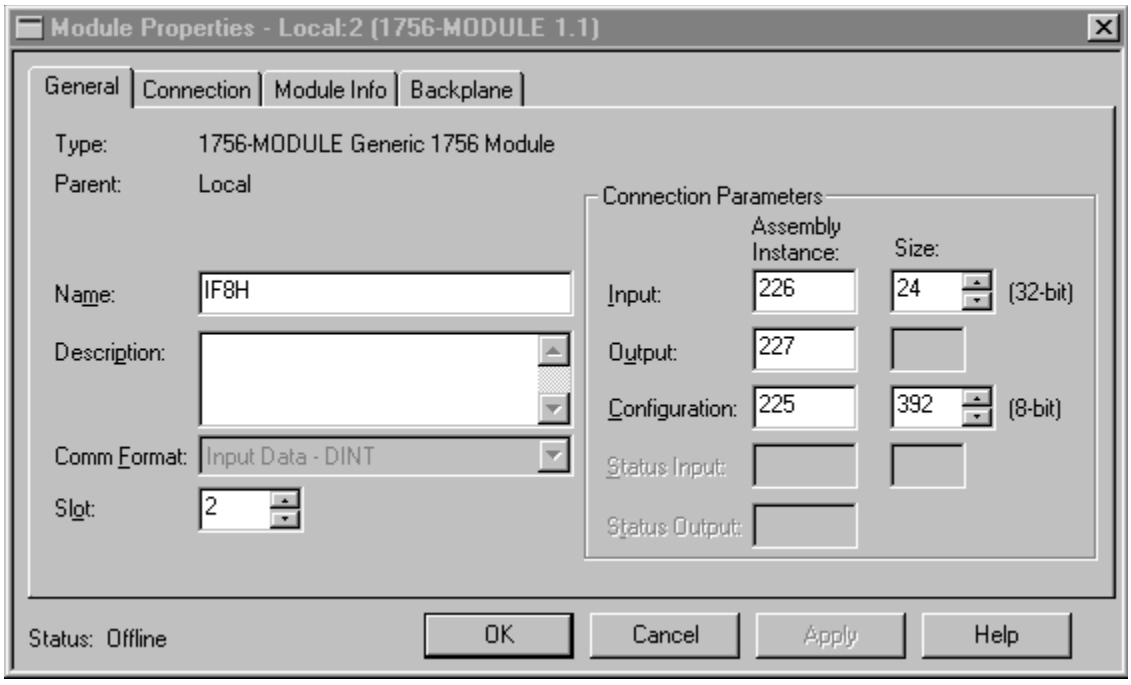
1. Open your project and go to the "Add I/O module" menu under controller configuration.

2. You will now see the list of all I/O modules. Select the "Generic 1756 I/O" option.

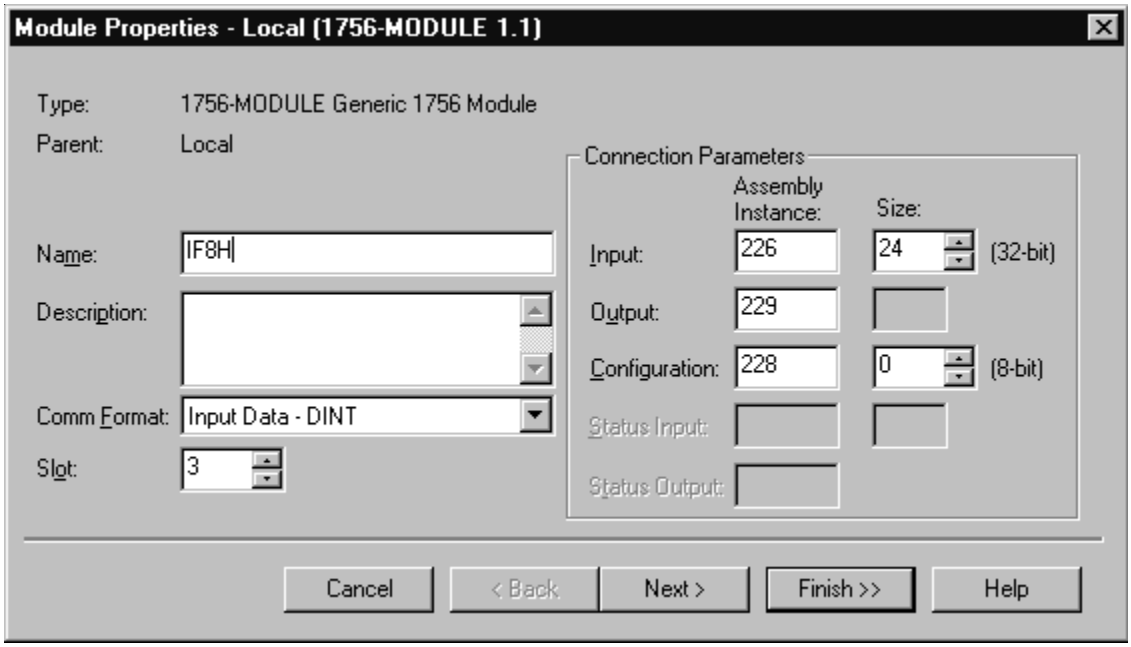


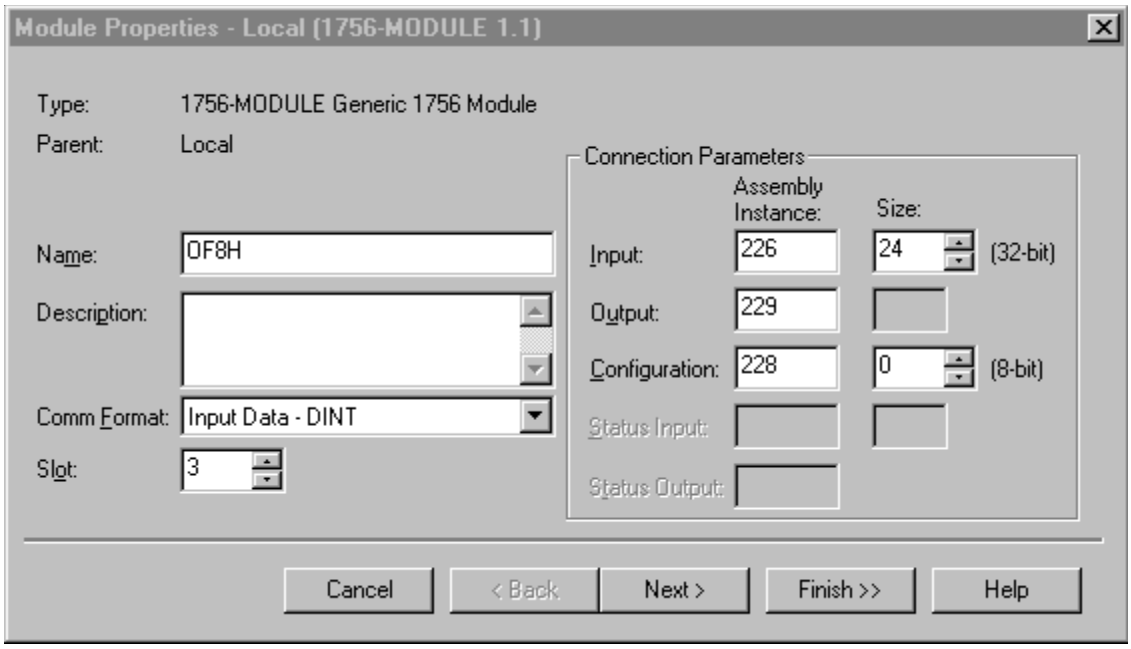
3. After clicking “OK” you are presented with the following dialog for setting up the general information about the module. Use the same values specified here:

Owner Controller Connection (Controller provides configuration)

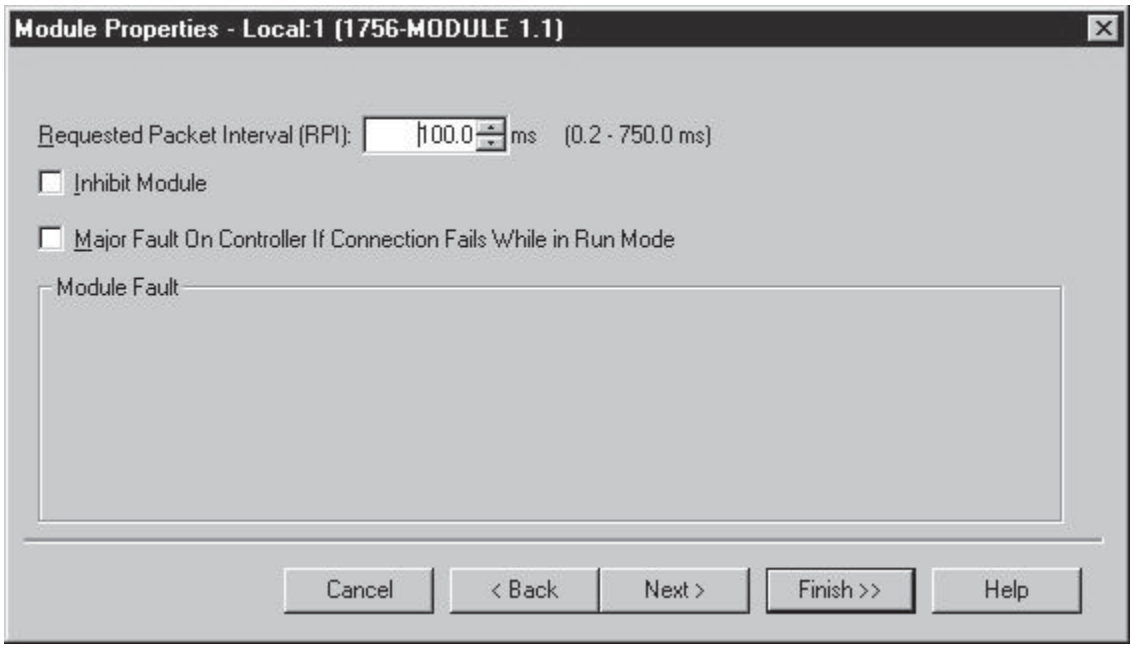


Listen-only controller connection. (Controller does not provide configuration but monitors input data only. Another owner controller must exist.)





4. Specify an RPI interval between 10.0 and 750.0 ms:



Appendix D

Additional HART Protocol Information

This chapter discussed the HART protocol and provides references for additional information about the protocol. This chapter provides:

- HART Protocol background information.
- Common Practice command sets.
- Extended command sets.
- References to additional information.

Overview

HART Field Communications Protocol is widely accepted in the industry as the standard for digitally enhanced 4-20mA communication with smart field instruments. The HART Protocol message structure, command set and status are discussed in this chapter.

The HART Command Set is organized into three groups and provides read/write access to a wide array of information available in smart field instruments.

Universal Commands provide access to information that is useful in normal plant operation such as the instrument manufacturer, model, tag, serial number, descriptor, range limits, and process variables. All HART devices must implement Universal Commands.

Common Practice Commands provide access to functions, which can be carried out by many devices though, not all.

Device Specific Commands provide access to functions that may be unique to a particular device.

Message Structure

This section describes the transaction procedure, character coding and message structure of the HART protocol. These correspond to layer 2 - the data-link layer - of the OSI protocol reference model.

Master-slave operation

HART is a master-slave protocol. This means that each message transaction is originated by the master; the slave (field) device only replies when it receives a command message addressed to it. The reply from the slave device acknowledges that the command has been received, and may contain data requested by the master.

Multiple master operation

The HART protocol allows for two active masters in a system, one primary and one secondary. The two masters have different addresses, therefore each can positively identify replies to its own command messages.

Transaction procedure

HART is a half-duplex protocol; after completion of each message, the FSK carrier signal must be switched off, to allow the other station to transmit. The carrier control timing rules state that the carrier should be turned on not more than 5 bit times before the start of the message (that is, the preamble) and turned off not more than 5 bit times after the end of the last byte of the message (the checksum).

The master is responsible for controlling message transactions. If there is no reply to a command within the expected time, the master should retry the message. After a few retries, the master should abort the transaction, since presumably the slave device or the communication link has failed.

After each transaction is completed, the master should pause for a short time before sending another command, to allow an opportunity for the other master to break in if it wishes. This way, two masters (if they are present) take turns at communicating with the slave devices. Typical message lengths and delays allow two transactions per second.

Burst mode

To achieve a higher data rate, some field devices implement an optional burst mode. When switched into this mode, a slave device repeatedly sends a data message, as though it had received a specific command to do so. Special commands (#107, #108, #109) are used to start and stop this mode of operation, and to choose which command should be assumed. (If burst mode is implemented, Commands #1, #2 and #3 must be supported; other commands are optional.) There is a short pause after each burst message, to allow a master device to send a command to stop the burst mode operation, or to initiate any other single transaction (after which burst messages will continue).

Generally, burst mode is only useful if there is just one field device attached to a pair of wires (since only one field device on a loop can be in burst mode at any one time). In burst mode, more than three messages can be transmitted per second.

Status

Two bytes of status, also known as the response code, are included in every message from a field (slave) device. These two bytes convey three different types of information: communication errors, command responses, and field device status.

If an error is detected in the outgoing communication, the most-significant bit (bit 7) of the first byte is set to 1, and the details of the error are reported in the rest of that byte; the second byte is then all zeros.

If no error is detected in the outgoing communication, bit 7 of the first byte is 0, the remainder of the byte contains the command response, indicating any problem with the received command, and the second byte contains the field device status, indicating the operational state of the slave device.

Communication errors are mostly those that would be detected by a HART: parity, overrun and framing errors. The field device also reports overflow of its receive buffer, and any discrepancy between the message content and the checksum received.

Command response codes (integers in the range 0 to 127) are categorized as either errors or warnings, and as having either a single meaning or multiple meanings

Universal Commands

Command		Data in Command			Data in Reply		
#	Function	Byte	Data	Type	Byte	Data	Type
0	Read unique identifier		None		0 1 2 3 4 5 6 7 8 9–11 * Bit 0 = multisensor device; Bit 1 = EEPROM control required; Bit 2 = protocol bridge device	254 (expansion) Manufacturer identification code Manufacturer device type code Number of preambles required Universal command revision Device-specific command revision Software revision Hardware revision Device function flags* Device ID number	 (H) (B)
1	Read primary variable				0 1–4	PV units code Primary variable	 (F)
2	Read current and percent of range		None		0–3 4–7	Current (mA) Primary variable	(F) (F)
3	Read current and four (predefined) dynamic variables		None		0–3 4 5–8 9 10–13 14 15–18 19 20–23 (truncated after last supported variable)	Current (mA) PV units code Primary variable SV units code Secondary variable TV units code Third variable FV units code Fourth variable	
6	Write polling address	0	Polling address			As in command	
11	Read unique identifier associated with tag	0-5	Tag	(A)	0-11	As Command 0	
12	Read message		None		0–23	Message (32 characters)	(A)
13	Read tag, descriptor, date		None		0–5 6–17 18–20	Tag (8 characters) Descriptor (16 characters) Date	(A) (A) (D)
14	Read PV sensor information		None		0–2 3 4–7 8–11 12–15	Sensor serial number Units code for sensor limits and minimum span Upper sensor limit Lower sensor limit Minimum span	 (F) (F) (F)

Command		Data in Command			Data in Reply		
#	Function	Byte	Data	Type	Byte	Data	Type
15	Read output information		None		0 1 2 3–6 7–10 11–14 15 16	Alarm select code Transfer function code PV/range units code Upper-range value Lower-range value Damping value (seconds) Write-protect code Private-label distributor code	 (F) (F) (F)
16	Read final assembly number		None		0–2	Final assembly number	
17	Write message	0-23	Message (32 characters)	(A)		As in command	
18	Write tag, descriptor, date	0–5 6–17 18–20	Tag (8 characters) Descriptor (16 characters) Date	(A) (A) (D)		As in command	
19	Write final assembly number	0–2	Final assembly number			As in command	

Common Practive Commands

Command		Data in Command			Data in Reply		
#	Function	Byte	Data	Type	Byte	Data	Type
33	Read transmitter variables	(truncated after last requested code	None		0 1 2–5 6 7 8–11 12 13 14–17 18 19 20–23 (truncated after last requested variable)	Transmitter variable code for slot 0 Units code for slot 0 Variable for slot 0 Transmitter variable code for slot 1 Units code for slot 1 Variable for slot 1 Transmitter variable code for slot 2 Units code for slot 2 Variable for slot 2 Transmitter variable code for slot 3 Units code for slot 3 Variable for slot 3	 (F) (F) (F) (F)
34	Write damping value	0–3	Damping value (seconds)	(F)		As in command	(F)
35	Write range values	0 1–4 5–8	Range units code Upper-range value Lower-range value		 (F) (F)	As in command	(F) (F)
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	0 *0 = burn EEPROM; 1 = copy EEPROM to RAM	EEPROM control code*			As in command	
40	Enter/exit fixed current mode	0–3 *0 = exit fixed current mode	Current (mA)*		(F)	As in command	
41	Perform device self-test		None			None	
42	Perform master reset		None			None	
43	Set (trim) PV zero		None			None	
44	Write PV units	0	PV units code			As in command	
45	Trim DAC zero	0-3	Measured current (mA)			As in command	
46	Trim DAC gain	0–3	Measured current (mA)		(F)	As in command	
47	Write transfer function	0	Transfer function code			As in command	
48	Read additional device status		None		0–5 6–7 8–10 11–13 14–24 *24 bits each	Device-specific status Operational modes Analog outputs saturated* Analog outputs fixed* Device-specific status LSB ... MSB refers to AO #1 ... #24.	(B) (B) (B) (B)

Command		Data in Command			Data in Reply		
#	Function	Byte	Data	Type	Byte	Data	Type
49	Write PV sensor serial number	0–2	Sensor serial number			As in command	
			number				
50	Read dynamic variable assignments		None		0	PV transmitter variable code	
					1	SV transmitter variable code	
					2	TV transmitter variable code	
					3	FV transmitter variable code	
51	Write dynamic variable assignments	0	PV transmitter variable code			As in command	
		1	SV transmitter variable code				
		2	TV transmitter variable code				
		3	FV transmitter variable code				
52	Set transmitter variable zero	0	Transmitter variable code			As in command	
53	Write transmitter variable units	0	Transmitter variable code			As in command	
		1	Transmitter variable units code				
54	Read transmitter variable information		Transmitter variable code		0	Transmitter variable code	
					1–3	Transmitter variable sensor serial number	
					4	Transmitter variable limits units code	
					5–8	Transmitter variable upper limit	(F)
					9–12	Transmitter variable lower limit	(F)
					13–16	Transmitter variable damping value (seconds)	(F)
					17–20	Transmitter variable minimum span	(F)
55	Write transmitter variable damping value	0	Transmitter variable code			As in command	
		1-4	Transmitter variable damping value (seconds)				
56	Write transmitter variable sensor serial number	0	Transmitter variable code			As in command	
		1-3	Transmitter variable sensor				
57	Read unit tag, descriptor, date		None		0–5	As in command	(A)
					6–17		(A)
					18–20		(D)
58	Write unit tag, descriptor, date	0–5	Unit tag (8 characters)	(A)		As in command	
		6–17	Unit descriptor (16 characters)	(A)			
		18–20	Unit date	(D)			
59	Write number of response preambles	0	Number of response preambles			As in command	
60	Read analog output and percent of range	0	Analog output number code		0	Analog output number code	
					1	Analog output units code	
					2–5	Analog output level	
					6–9	Analog output percent of range	

Command		Data in Command			Data in Reply		
#	Function	Byte	Data	Type	Byte	Data	Type
61	Read dynamic variables and PV analog output		None		0 1–4 5 6–9 10 11–14 15 16–19 20 21–24	PV analog output units code PV analog output level PV units code Primary variable SV units code Secondary variable TV units code Tertiary variable FV units code Fourth variable	 (F) (F) (F) (F) (F)
62	Read analog outputs	0 1 2 3 (truncated after last requested code)	Analog output number; code for slot 0 Analog output number; code for slot 1 Analog output number; code for slot 2 Analog output number; code for slot 3		0 1 2–5 6 7 8–11 12 13 14–17 18 19 20–23 (truncated after last requested level)	Slot 0 analog output number code Slot 0 units code Slot 0 level Slot 1 analog output number code Slot 1 units code Slot 1 level Slot 2 analog output number code Slot 2 units code Slot 2 level Slot 3 analog output number code Slot 3 units code Slot 3 level	 (F) (F) (F) (F)
63	Read analog output information	0	Analog output number code		0 1 2 3 4–7 8–11 12–15	Analog output number code Analog output alarm select code Analog output transfer function code Analog output range units code Analog output upper-range value Analog output lower-range value Analog output additional damping value (sec)	 (F) (F) (F)
64	Write analog output additional damping value	0 1-4	Analog output number code Analog output additional damping value (sec)	 (F)		As in command	
65	Write analog output range value	0 1 2-5 6-9	Analog output number code Analog output range units code Analog output upper-range value Analog output lower-range value	 (F) (F)		As in command	

Command		Data in Command			Data in Reply		
#	Function	Byte	Data	Type	Byte	Data	Type
66	Enter/exit fixed analog output mode	0 1 2–5	Analog output number code Analog output units code Analog output level*	 (F)		As in command	
		* "not a number" exits fixed output mode					
67	Trim analog output zero	0 1 2–5	Analog output number code Analog output units code Externally measured analog output level	 (F)		As in command	
68	Trim analog output gain	0 1 2–5	Analog output number code Analog output units code Externally measured analog output level	 (F)		As in command	
69	Write analog output transfer function	0 1	Analog output number code Analog output transfer function code			As in command	
70	Read analog output endpoint values	0	Analog output number code		0 1 2–5 6–9	Analog output number code Analog output endpoint units code Analog output upper endpoint value Analog output lower endpoint value	
107	Write burst mode transmitter variables (for Command #33)	0 1 2 3	Transmitter variable code for slot 0 Transmitter variable code for slot 1 Transmitter variable code for slot 2 Transmitter variable code for slot 3			As in command	
108	Write burst mode command number	0	Burst mode command number			As in command	
109	Burst mode control	0	Burst mode control code (0 = exit, 1 = enter)			As in command	
110	Read all dynamic variables		None		0 1–4 5 6–9 10 11–14 15 16–19	PV units code PV value SV units code SV value TV units code TV value FV units code FV value	 (F) (F) (F) (F)

Status

Two bytes of status, also called the response code, are included in every reply message from a field or slave device. These two bytes convey three types of information:

Communication errors
Command response problems
Field device status

If an error is detected in the outgoing communication, the most significant bit (bit 7) of the first byte is set to 1 and the details of the error are reported in the rest of that byte. The second byte is then all zeros.

If no error is detected in the outgoing communication, bit 7 of the first byte is 0 and the remainder of the byte contains the command response, which indicates any problem with the received command. The second byte contains status information pertaining to the operational state of the field or slave device.

Communication errors are typically those that would be detected by a UART (i.e., parity overrun and framing errors). The field device also reports overflow of its receive buffer and any discrepancy between the message content and the checksum received.

Response Codes

Table 3a (Communication Status / First Byte)

Bit 7 = 1: Communication 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)

Table 3b (Communication Status / First Byte)

Bit 7 = 0: Command Errors	
Bit 6-0 (not bit-mapped):	
0	No command specific error
1	(undefined)
2	Checksum Error
3	(Reserved)
4	RX buffer overflow
5	(undefined)
6	Transmitter-specific command error
7	In write-protect mode
8-15	Command-specific errors See table below
16	Access restricted
32	Device is busy
64	Command not implemented

Table 3c (First Byte)

	Description
Bit 0	All 0
Bit 1	
Bit 2	
Bit 3	
Bit 4	
Bit 5	
Bit 6	
Bit 7	

Table 3d (Second Byte)

	Description
Bit 0	Primary variable out of limits
Bit 1	Non-Primary variable out of limits
Bit 2	Analog output #1 saturated
Bit 3	Analog output #1 fixed
Bit 4	More status available
Bit 5	Cold start
Bit 6	Configuration changed
Bit 7	Field device malfunction

Note: Hexadecimal equivalents are quoted assuming only a single bit is set. In reallity, several bits may be set simultaneously, and the hex digits can be used together.

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

Getting Technical Assistance

If you need technical assistance, please review the information in Chapter 9, "Testing Your Module," before calling your local distributor of Spectrum Controls.

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 - 440-646-6900
United Kingdom - 01908 635230
Australia - 800-809-929 or (61) 398-990-335
Brazil - (55) 11 3618 8800
Europe - (49) 2104 960 333

Declaration of Conformity

Declaration available upon request.

Publication 0300196-02 Rev. D April 2004. Printed in U.S.A.

Corporate Headquarters
Spectrum Controls Inc.
P.O. Box 5533
Bellevue, WA 98006 USA
Fax: 425-641-9473
Tel: 425-746-9481

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

SPECTRUM
CONTROLS