User’s Manual Pub. 0300217-04 Rev. A

1769 4-Channel Isolated Analog HART Output Module

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

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

Notice

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

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

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

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

Limited Warranty

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

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

## Important Notes

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Chapter 1: Module Overview</td>
</tr>
<tr>
<td>2-1</td>
<td>Chapter 2: Installation and Wiring</td>
</tr>
<tr>
<td>3-1</td>
<td>Chapter 3: Configuring the OF4IH for CompactLogix Using RSLogix 5000</td>
</tr>
<tr>
<td>4-1</td>
<td>Chapter 4: Configuring the OF4IH for MicroLogix 1500 Using RSLogix 500</td>
</tr>
</tbody>
</table>

## Chapter 1 Module Overview

1.1 General Description
2.1 Output Types and Ranges
3.1 Data Formats
4.1 Hardware Features
5.1 System Overview
6.1 Module Power-up
7.1 Module Operation

## Chapter 2 Installation and Wiring

1.1 Required Tools and Equipment
2.1 Compliances to European Union Directives
3.1 CE Safety
4.1 Power Requirements
5.1 Considerations
6.1 Mounting
7.1 Panel Mounting
8.1 Wiring the Module
9.1 Wiring the Module to Check if the Module is Operating Correctly

## Chapter 3 Configuring the OF4IH for CompactLogix Using RSLogix 5000

1.1 Setting up the Generic Profile
2.1 Using the Add-On Profile
3.2 Installing the Add-On Profile
4.2 Adding the OF4IH Module to Your Logix Project
5.2 User Defined Data Types
6.2 Project Tags
7.2 Sample Project Ladder

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

1.2 Module Addressing
2.2 Configuring the 1769sc-OF4IH in a MicroLogix 1500 System
3.2 Using the Ladder Sample
4.3 Copying Subroutines from the Sample Project

### CHAPTER 5 MODULE DATA, STATUS, AND CHANNEL CONFIGURATION ............................................. 5-1

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>MODULE MEMORY MAP</td>
<td>5-1</td>
</tr>
<tr>
<td>5.2</td>
<td>ACCESSING INPUT IMAGE FILE DATA</td>
<td>5-2</td>
</tr>
<tr>
<td>5.3</td>
<td>INPUT DATA FILE</td>
<td>5-3</td>
</tr>
<tr>
<td>5.3.1</td>
<td>General Status Bits S0 to S3 (Word 0)</td>
<td>5-3</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Out of Service Status Bits OS0 to OS3 (Word 0)</td>
<td>5-3</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Over-Range Flag Bits O0 to O3 (Word 1)</td>
<td>5-4</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Under-Range Flag Bits U0 to U3 (Word 1)</td>
<td>5-4</td>
</tr>
<tr>
<td>5.3.5</td>
<td>Hold Last State Bits H0 to H3 (Word 1)</td>
<td>5-4</td>
</tr>
<tr>
<td>5.3.6</td>
<td>Channel X Command Value Echo (Words 2 to 5)</td>
<td>5-4</td>
</tr>
<tr>
<td>5.3.7</td>
<td>HART Packet Data (Words 6 to 25)</td>
<td>5-5</td>
</tr>
<tr>
<td>5.3.8</td>
<td>Message Slave Control (Word 26)</td>
<td>5-5</td>
</tr>
<tr>
<td>5.3.9</td>
<td>Message Reply Size (Word 27)</td>
<td>5-5</td>
</tr>
<tr>
<td>5.3.10</td>
<td>Message Reply Buffer (Words 28...47)</td>
<td>5-5</td>
</tr>
<tr>
<td>5.3.11</td>
<td>Reserved (Words 48...69)</td>
<td>5-5</td>
</tr>
<tr>
<td>5.5</td>
<td>MODULE CONFIGURATION</td>
<td>5-5</td>
</tr>
<tr>
<td>5.3.12</td>
<td>Channel X General Settings (Words 0, 8, 16, 24)</td>
<td>5-9</td>
</tr>
<tr>
<td>5.3.13</td>
<td>Channel X Output Type and Data Format (Words 1, 9, 17, 25)</td>
<td>5-12</td>
</tr>
<tr>
<td>5.3.14</td>
<td>Fault Value (Words 2, 10, 18, 26)</td>
<td>5-14</td>
</tr>
<tr>
<td>5.3.15</td>
<td>Program/Idle Value (Words 3, 11, 19, 27)</td>
<td>5-14</td>
</tr>
<tr>
<td>5.3.16</td>
<td>Low Clamp Value (Words 4, 12, 20, 28)</td>
<td>5-14</td>
</tr>
<tr>
<td>5.3.17</td>
<td>High Clamp Value (Words 5, 13, 21, 29)</td>
<td>5-15</td>
</tr>
<tr>
<td>5.3.18</td>
<td>Ramp Rate Value (Words 6, 14, 22, 30)</td>
<td>5-15</td>
</tr>
<tr>
<td>5.3.19</td>
<td>Spare (Words 7, 15, 23, 31)</td>
<td>5-16</td>
</tr>
<tr>
<td>5.3.20</td>
<td>Pad (Word 32)</td>
<td>5-16</td>
</tr>
<tr>
<td>5.3.21</td>
<td>HART Configuration Word (Word 33)</td>
<td>5-16</td>
</tr>
<tr>
<td>5.3.22</td>
<td>Channel X HART Slot Variables 0 &amp; 1 (Words 34, 36, 38, 40)</td>
<td>5-18</td>
</tr>
<tr>
<td>5.3.23</td>
<td>Channel X HART Slot Variables 2 &amp; 3 (Words 35, 37, 39, 41)</td>
<td>5-18</td>
</tr>
<tr>
<td>5.4</td>
<td>OUTPUT DATA FILE</td>
<td>5-18</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Channel X Command Value</td>
<td>5-19</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Unlatch Process High Alarms UH0 to UH3 (Word 4)</td>
<td>5-19</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Unlatch Process Low Alarms UL0 to UL3 (Word 4)</td>
<td>5-19</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Hart Suspend HS0 to HS3 (Word 4)</td>
<td>5-20</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Packet Just Scanned (Word 5)</td>
<td>5-20</td>
</tr>
<tr>
<td>5.4.6</td>
<td>Message Master Control (Word 6)</td>
<td>5-21</td>
</tr>
<tr>
<td>5.4.7</td>
<td>Message Request Size (Word 7)</td>
<td>5-21</td>
</tr>
<tr>
<td>5.4.8</td>
<td>Message Request Buffer (Words 8...27)</td>
<td>5-21</td>
</tr>
<tr>
<td>5.4.9</td>
<td>Reserved (Words 28...49)</td>
<td>5-21</td>
</tr>
<tr>
<td>5.5</td>
<td>DETERMINING MODULE UPDATE TIME</td>
<td>5-21</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Module Update Time</td>
<td>5-21</td>
</tr>
</tbody>
</table>

### CHAPTER 6 ENABLING AND USING HART ON THE 1769SC-OF4IH .................................................. 6-1

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>CONFIGURING THE MODULE FOR HART</td>
<td>6-1</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Configuring the OF4IH Module for (Hart Acquisition/Communication)</td>
<td>6-1</td>
</tr>
<tr>
<td>6.2</td>
<td>HART PACKET DATA</td>
<td>6-2</td>
</tr>
<tr>
<td>6.2.1</td>
<td>How the Module Connects to a Field Device</td>
<td>6-2</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Auto Acquisition</td>
<td>6-3</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Packet Interval</td>
<td>6-11</td>
</tr>
<tr>
<td>6.3</td>
<td>SENDING AND RECEIVING MESSAGES</td>
<td>6-12</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Module Output Tags Used for Messaging</td>
<td>6-12</td>
</tr>
</tbody>
</table>
CHAPTER 7 PROGRAMMING EXAMPLES ................................................................. 7-1

SECTION 7.1 COMPACTLOGIX ........................................................................ 7-1
  7.1.1 Reset/Reconfig ...................................................................................... 7-1
  7.1.3 Swap Byte Order .................................................................................. 7-3
  7.1.4 Converting Unpacked ASCII to Packed ASCII ..................................... 7-3

SECTION 7.2 MICROLOGIX 1500 ...................................................................... 7-6
  7.2.1 MAIN Routine ...................................................................................... 7-7
  7.2.2 PACKETS Routine ................................................................................ 7-7
  7.2.3 MSG_TO_MOD Routine ...................................................................... 7-11
  7.2.4 SRC_CHECK Routine ........................................................................... 7-28
  7.2.5 DEST_CHECKSUM Routine ................................................................. 7-30
  7.2.6 HART_MSG Routine .......................................................................... 7-32
  7.2.7 WORD_BYTE Routine ...................................................................... 7-43
  7.2.8 BYTE_WORD Routine ........................................................................ 7-48

CHAPTER 8 DIAGNOSTICS AND TROUBLESHOOTING ........................................... 8-1

SECTION 8.1 SAFETY CONSIDERATIONS .......................................................... 8-1
  8.1.1 Indicator Lights ................................................................................... 8-1
  8.1.2 Stand Clear of Equipment .................................................................. 8-1
  8.1.3 Program Alteration ............................................................................ 8-1
  8.1.4 Safety Circuits ................................................................................... 8-2

SECTION 8.2 MODULE OPERATION VS. CHANNEL OPERATION ......................... 8-2

SECTION 8.3 POWER-UP DIAGNOSTICS .......................................................... 8-2

SECTION 8.4 CHANNEL DIAGNOSTICS ......................................................... 8-3
  8.4.1 Invalid Channel Configuration Detection ........................................... 8-3
  8.4.2 Over or Under-Range Detection .......................................................... 8-3

SECTION 8.5 NON-CRITICAL VS. CRITICAL MODULE ERRORS ....................... 8-3

SECTION 8.6 MODULE ERROR DEFINITION TABLE ......................................... 8-3
  8.6.1 Module Error Field ............................................................................ 8-4
  8.6.2 Extended Error Information Field ....................................................... 8-4

SECTION 8.7 ERROR CODES ............................................................................ 8-5

SECTION 8.8 MODULE INHIBIT FUNCTION .................................................... 8-6
Preface

NOTE

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

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

- Who should use this manual
- How to use this manual
- Related publications
- Conventions used in this manual
- Rockwell Automation support

Who Should Use This Manual

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

How to Use This Manual

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

Related Documentation

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

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

<table>
<thead>
<tr>
<th>Wiring and grounding information</th>
<th>Refer to this Document</th>
<th>Allen-Bradley Pub. No.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allen-Bradley Programmable Controller Grounding and Wiring Guidelines</td>
<td>1770-4.1</td>
</tr>
</tbody>
</table>

Technical Support

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

- USA 440-646-6900
- United Kingdom 01908 635230
- Australia 1800-809-929
- Mexico 001-888-365-8677
- Brazil (55) 11 3618 8800
- Europe +49 211 41553 63

or send an email to support@spectrumcontrols.com

Documentation

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

Conventions Used in This Manual

The following conventions are used throughout this manual:

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

<table>
<thead>
<tr>
<th>WARNING</th>
<th>Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. These messages help you to identify a hazard, avoid a hazard, and recognize the consequences.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTENTION</td>
<td>Actions ou situations risquant d’entraîner des blessures pouvant être mortelles, des dégâts matériels ou des pertes financières. Les messages « Attention » vous aident à identifier un danger, à éviter ce danger et en discerner les conséquences.</td>
</tr>
<tr>
<td>NOTE</td>
<td>Identifies information that is critical for successful application and understanding of the product.</td>
</tr>
</tbody>
</table>
Chapter 1
Module Overview

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

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

Section 1.1
General Description

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

Section 1.2
Output Types and Ranges

The table below lists the output ranges for the module.

Table 1-1 Current Output Ranges

<table>
<thead>
<tr>
<th>Current Output Range</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 20 mA</td>
<td></td>
</tr>
<tr>
<td>4 to 20 mA</td>
<td></td>
</tr>
</tbody>
</table>

Section 1.3
Data Formats

For each channel the data can be configured for:

- Engineering Units ×1
- Scaled-for-PID
- Percent of full scale
- Raw/proportional counts
Section 1.4 
Hardware Features

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

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

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

1.4.1 General Diagnostic Features

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

Section 1.5 
System Overview

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

1.5.1 Module Power-Up

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

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

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

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

### 1.5.2 Module Operation

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

<table>
<thead>
<tr>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>The HART data is sent and received asynchronously from the analog acquisition process and therefore does not directly affect the analog update time.</td>
</tr>
</tbody>
</table>

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

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

See the block diagram below.

![Block Diagram](image)

**Figure 1-1 1769sc-OF4IH Block Diagram**
Chapter 2
Installation and Wiring

This chapter explains how to:
- Tools and Equipment
- Compliance to European Union directives
- Power requirements
- General considerations
- Mounting

Section 2.1
Required Tools and Equipment

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

Section 2.2
Compliance to European Union Directives

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

2.2.1 EMC Directive

The 1769sc-OF4IH module 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 50082-2 EMC–Generic Immunity Standard, Part 2 - Industrial Environment

This product is intended for use in an industrial environment.
2.2.2 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, 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 B113

Section 2.3
CE Safety
This product is designed to, and verified in compliance with, European Union Safety Standards:
- EN61131-2
- EN61010-1

Section 2.4
Power Requirements
You must ensure that your power supply has sufficient current output to support your system configuration. The module receives power through the bus interface from the +5 VDC/+24 VDC system power supply. The maximum current drawn by the module is shown in the table below:

<table>
<thead>
<tr>
<th>5 VDC</th>
<th>24 VDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>180 mA</td>
<td>200 mA</td>
</tr>
</tbody>
</table>

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

Section 2.5
Considerations
Compact I/O is suitable for use in an industrial environment when installed in accordance with these instructions. Specifically, this equipment is intended for use in clean, dry environments (Pollution degree 2\(^1\) and to circuits not exceeding Over Voltage Category II\(^2\) (IEC 60664-1)\(^3\))

2.5.1 Hazardous Location Considerations
This equipment is suitable for use in Class I, Division 2, Groups A, B, C, D or

---

\(^1\) Pollution Degree 2 is an environment where, normally, only non-conductive pollution occurs except that occasionally a temporary conductivity caused by condensation shall be expected.

\(^2\) Over Voltage Category II is the load level section of the electrical distribution system. At this level, transient voltages are controlled and do not exceed the impulse voltage capability of the product’s insulation.

\(^3\) Pollution Degree 2 and Over Voltage Category II are International Electrotechnical Commission (IEC) designations.
non-hazardous locations only. The following WARNING statement applies to use in hazardous locations.

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

2.5.2 Prevent Electrostatic Discharge

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

2.5.3 Remove Power

<table>
<thead>
<tr>
<th>WARNING</th>
<th>Remove power before removing or inserting this module. When you remove, or insert, a module with power applied, an electrical arc may occur. An electrical arc can cause personal injury or property damage by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Warning Icon]</td>
<td>• Sending an erroneous signal to your system’s field devices, causing unintended machine motion.</td>
</tr>
<tr>
<td>![Warning Icon]</td>
<td>• Causing an explosion in a hazardous environment.</td>
</tr>
<tr>
<td>![Warning Icon]</td>
<td>• Causing an electrical arc. Such arcing causes excessive wear to contacts on both the module and its mating connector, and may lead to premature failure.</td>
</tr>
</tbody>
</table>
2.5.4 Selecting a Location

Reducing Noise
Most applications require installation in an industrial enclosure to reduce the effects of electrical interference. Analog inputs are highly susceptible to electrical noise. Electrical noise coupled to the analog inputs will reduce the performance (accuracy) of the module.
Group your modules to minimize adverse effects from radiated electrical noise. Consider the following conditions when selecting a location for the analog module. Position the module:

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

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

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

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

<table>
<thead>
<tr>
<th>Module Catalog</th>
<th>Max Thermal Dissipation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1769-ADN</td>
<td>4.7</td>
</tr>
<tr>
<td>1769-HSC</td>
<td>6.2</td>
</tr>
<tr>
<td>1769-IQ32</td>
<td>4.8</td>
</tr>
<tr>
<td>1769-OW16</td>
<td>4.8</td>
</tr>
<tr>
<td>1769-OB32</td>
<td>4.5</td>
</tr>
</tbody>
</table>

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

Power Supply Distance
You can install as many modules as your power supply can support. However, all 1769 I/O modules have a power supply distance rating. The maximum I/O module rating is 8, which means that a module may not be located more than 8 modules away from the system power supply.
Section 2.6 Mounting

**WARNING**

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

### 2.6.1 Minimum Spacing

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

**Figure 2-2. Minimum Spacing**
Figure 2-3. Module Parts List

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

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

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

**Figure 2-4. Module Assembly**

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

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

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

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

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

8. Lock the end cap bus terminator (6).

**WARNING**

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

Section 2.7 Panel Mounting

**WARNING**

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

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

**Panel Mounting Using the Dimensional Template**

**Figure 2-5. Dimensional Template**

For more than 2 modules: (number of modules - 1) x 35 mm (1.38 in.)

Refer to controller documentation for this dimension

![Dimensional Template Diagram](image)

**Panel Mounting Procedure Using Modules as a Template**

The following procedure allows you to use the assembled modules as a template.
for drilling holes in the panel. If you have sophisticated panel mounting equipment, you can use the dimensional template provided on the previous page. Due to module mounting hole tolerance, it is important to follow these procedures:

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

NOTE

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

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

**DIN Rail Mounting**

The module can be mounted using the following DIN rails:

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

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

**2.7.1 Replacing a Single Module within a System**

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

NOTE

It may be necessary to rock the module slightly from front to back to remove it, or, in a panel-mounted system, to loosen the screws of adjacent modules.
6. Before installing the replacement module, be sure that the bus lever on the module to be installed and on the right-side adjacent module or end cap are in the unlocked (fully right) position.

7. Slide the replacement module into the open slot.

8. Connect the modules together by locking (fully left) the bus levers on the replacement module and the right-side adjacent module.

9. Replace the mounting screws (or snap the module onto the DIN rail).

Section 2.8
Wiring the Module

When wiring your system, use the following guidelines:

General
- Channels are isolated from one another by ±500 VDC maximum.
- As a general rule, allow at least 15.2 cm (6 in.) of separation for every 120 V of power.
- Routing field wiring in a grounded conduit can reduce electrical noise.
- If field wiring must cross AC or power cables, ensure that they cross at right angles.
- Provision shall be made to prevent the rated voltage being exceeded by the transient disturbances of more than 140%.
- The equipment shall be installed in an enclosure that provides a degree of protection not less than IP 54 in accordance with EN 60079-0 and used in an environment of not more than pollution degree 2. The enclosure shall be accessible only with the use of a tool.
- Grounding to earth is accomplished through mounting of modules on rail.
- Subject devices are for operation in Ambient Temperature Range: 0 °C to +60 °C.

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

Grounding
- This product is intended to be mounted to a well-grounded mounting surface such as a metal panel. Additional grounding connections from the module’s mounting tabs or DIN rail (if used) are not required unless the mounting surface cannot be grounded.
- Keep cable shield connections to ground as short as possible.
• Ground the shield drain wire at one end only. The typical location is as follows.
  - If it is necessary to connect the shield drain wire at the module end, connect it to earth ground using a panel or DIN rail mounting screw.
  - Refer to Industrial Automation Wiring and Grounding Guidelines, Allen-Bradley publication 1770-4.1, for additional information.

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

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

Figure 2-6. Terminal Block
To remove the terminal block, loosen the upper and lower retaining screws. The terminal block will back away from the module as you remove the screws. When replacing the terminal block, torque the retaining screws to 0.46 Nm (4.1 in-lbs).

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

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

3. Tighten the terminal screw making sure the pressure plate secures the wire. Recommended torque when tightening terminal screws is 0.68 Nm (6 in-lbs).
NOTE
If you need to remove the finger-safe cover, insert a screwdriver into one of the square, wiring holes and gently pry the cover off. If you wire the terminal block with the finger-safe cover removed, you may not be able to put it back on the terminal block because the wires will be in the way.

Wire Size and Terminal Screw Torque
Each terminal accepts up to two wires with the following restrictions:

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

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

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

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

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

Figure 2-7
To wire your module follow these steps.
1. At each end of the cable, strip some casing to expose the individual wires.
WARNING

HAZARD OF DAMAGE TO EQUIPMENT

Be careful when stripping wires. Wire fragments that fall into a module could cause damage at power up.

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

Wiring Diagram

Figure 2-8. Wiring Diagram

Calibration

The isolated HART module is initially calibrated at the factory.
**2.8.2 Perform the Startup Procedure**

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

<table>
<thead>
<tr>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the module status LED does not turn on, cycle power. If the condition persists, contact your local distributor or Spectrum Controls for assistance.</td>
</tr>
</tbody>
</table>

**2.8.3 Monitor Module Status to Check if the Module is Operating Correctly**

Module and channel configuration errors are reported to the controller. These errors are typically reported in the controller’s I/O status file. Channel status data is also reported in the module’s input data table, so these bits can be used in your control program to flag a channel error.
Chapter 3
Configuring the OF4IH for CompactLogix Using RSLogix 5000

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

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

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

Section 3.1
Setting up the Generic Profile

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

- Data Types
  - User-Defined
  - Strings
  - Predefined
  - Module-Defined

- I/O Configuration
  - [1] 1769-L35E Ethernet Port Local
  - [1] CompactBus Local
  - [1] 1769-MODULE_OF4IH0
1. Click on the new project icon or on the File pull-down menu and select New. The following dialog appears:

![New Controller dialog](image)

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

![Main RSLogix 5000 dialog](image)
3. In the Controller Organizer on the left of the dialog, right click [0]CompactBus Local, and select New Module.

The following dialog appears:

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

The following default Generic Profile dialog appears:

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

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

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

<table>
<thead>
<tr>
<th>1769 I/O Module</th>
<th>Comm Format</th>
<th>Parameter</th>
<th>Assembly Instance</th>
<th>Size (16-Bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OF4IH</td>
<td>Data-INT</td>
<td>Input</td>
<td>101</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Config</td>
<td>102</td>
<td>42</td>
</tr>
</tbody>
</table>

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

![Module Properties - Local:1 (1769 MODULE 1.1)](image)

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

### Section 3.2 Using The Add-On Profile

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

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

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

3.2.2 Adding the OF4IH Module to Your Logix Project

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

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

2. When the dialog opens, select the By Vendor tab, and expand the Spectrum Controls folder.
3. Highlight the module and click OK.
5. Configure the module using the custom configuration dialogs:

![Configuration dialog](image)

**NOTE**

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

---

**Section 3.3 User Defined Data Types**

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

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

**Sample Project**

- Data Types
- User-Defined
  - ConfigurationStructure
  - InputStructure
  - OF4IHMessage
  - OF4IHPassThruMsg
  - OutputStructure
  - PACKET0
  - PACKET1
  - PACKET2
  - PACKET3
  - PACKET4

**Your Project**

Drag and drop one at a time

---

User's Manual 0300217-04 Rev. A
NOTE

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

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

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

### Section 3.4 Project Tags

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

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

\(^1\) Refer to Chapter 6 for more details.
\(^2\) Refer to Chapter 7 for more details.
After copying the controller tags you can copy the program tags next. Follow the same procedure shown in Figure 4-8.

The figures below show examples of the configuration tags, input tags and output tags. Refer to Chapter 5 for information on how to configure the module and or reading the input data.
Refer to Chapter 6 for information regarding HART packet tags and pass-through tags:

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value</th>
<th>Input Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4IHConfiguration DIOConfigWord0</td>
<td>2#1000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOConfigWord1</td>
<td>2#0000_0001_0000_0001</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOEnableValue</td>
<td>0000</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOLowClamp</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOHighClamp</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIORampRate</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOReserved</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOConfigWord2</td>
<td>2#0000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOConfigWord3</td>
<td>2#0000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOEnableValue</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOReserved</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOConfigWord2</td>
<td>2#0000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOConfigWord3</td>
<td>2#0000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOEnableValue</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOReserved</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOConfigWord2</td>
<td>2#0000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOConfigWord3</td>
<td>2#0000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOEnableValue</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOReserved</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOConfigWord2</td>
<td>2#0000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOConfigWord3</td>
<td>2#0000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOEnableValue</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHConfiguration DIOReserved</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
</tbody>
</table>

**Figure 3-2. Input Tags**

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
<th>Input Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4IHInput StatusWord0</td>
<td>2#0000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHInput StatusWord1</td>
<td>2#0000_0000_0000_0000</td>
<td>Binary INT</td>
</tr>
<tr>
<td>D4IHInput DiData</td>
<td>0000</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHInput DiData</td>
<td>-32767</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHInput DiData</td>
<td>-32767</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHInput HiData</td>
<td>0000</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHInput HiData</td>
<td>{...}</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHInput ModSlaveControl</td>
<td>16#C1C1</td>
<td>Hex INT</td>
</tr>
<tr>
<td>D4IHInput ModResponseSize</td>
<td>0</td>
<td>Decimal INT</td>
</tr>
<tr>
<td>D4IHInput ModResponseBuffer</td>
<td>{...}</td>
<td>Decimal INT</td>
</tr>
</tbody>
</table>
Section 3.5
Sample Project Ladder

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

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

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

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

1. Select the program or routine.
2. Right mouse click and select copy.
3. Go to your project and select the appropriate program or task to place the new routine or program.
4. Right mouse click and select paste as shown below:

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

1. Open the routine that contains the ladder you want to copy.
2. Select the rungs to copy.
3. Right mouse click and select copy.
4. Open the routine in your project where you wish to paste the new rungs.
5. Right mouse click and select paste as shown below:
Chapter 4 Configuring the OF4IH for a MicroLogix 1500 Using RSLogix 500

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

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

Section 4.1 Module Addressing

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

Figure 4-1. Module Memory Map
For example, to obtain the general status for channel 2 of the module located in slot e, use address I:e.0/2.

**Figure 4-2. Address Example**

[Diagram showing address elements: Slot, Word, Bit, Input File Type, Element Delimiter, Word Delimiter, Bit Delimiter, I:e.0/2]

| NOTE | The end cap does not use a slot address. |

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

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

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

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

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

3. This dialog allows you to manually enter expansion modules into expansion slots, or to automatically read the configuration of the controller. To read the existing controller configuration, click on the **Read IO Config** button.
4. A communications dialog appears, identifying the current communications configuration so that you can verify the target controller. If the communication settings are correct, click on **Read IO Config**:

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

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

![Module Configuration Dialog](image)

**NOTE**

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

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

![Generic Extra Data Configuration](image)

For a complete description of each of these parameters and the choices available for each of them, refer to Chapter 5.
Section 4.3
Using the Ladder Sample

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

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

Table 4-1. Ladder Routines

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

You have the choice to either use the sample project or copy and past the pieces you need from the project.
4.3.1 Copying Subroutines from the Sample Project

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

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

Figure 4-2. Copying Routines

4.3.2 Copying Ladder from the Sample Project

To copy ladder, follow the procedure below:

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

4.3.3 Importing Tag Database and Rung Comments

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

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

3. Select the RSLogix 500 tab and click OK.
4. Select the location for the export file.
5. In your project, from the Tools menu, select Database, and select ASCII Import. See image below:
After selecting ASCII import the following dialog should appear:

7. Select the RSLogix 500 radio button and leave everything else at default. After making your selections, press the OK button.

8. Select the export file from steps 4 and 5 and click Open. You may be prompted for multiple files depending on the selections you made in step 7.
Chapter 5
Module Data, Status, and Channel Configuration

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

This chapter contains information on the following:

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

Section 5.1
Module Memory Map

The module uses forty-eight input words for data and status bits (input image), twenty-eight output words, and forty-two configuration words.
Figure 5-1. Module Memory Map

- Input Image File
- Configuration File
- Output File

Input Image
70 Words

Configuration
48 Words

Output
50 Words

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

Section 5.2
Accessing Input Image File Data

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

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

Section 5.3
Input Data File

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

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

5.3.1 General Status Bits S0 to S3 (Word 0)

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

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

Bits OS0 through OS3 of word 0 indicate whether the associated channel is out of service (i.e. automatic HART acquisition is suspended).

NOTE

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

The general status bits S0 to S3 also indicate whether, or not, the input data for a particular channel, 0 through 3, is being properly converted (valid) by the

---

1 Changing bit values is not supported by all controllers. Refer to your controller manual for details.
module. This “invalid data” condition can occur (bit set) when the download of a new configuration to a channel is accepted by the module (proper configuration) but before the A/D converter can provide valid (properly configured) data to the 1769 bus master/controller. The following information highlights the bit operation of the Data Not Valid condition.

5.3.3 Over-Range Flag Bits O0 to O3 (Word 1)
Over-range bits for channels 0 through 3 are contained in word 1. They apply to all output types. When set (1), the over-range flag bit indicates an output signal that is at the maximum of its normal operating range for the represented channel or sensor. The module automatically resets (0) the bit when the data value falls below the maximum for that range.

| NOTE | This bit can also be set if the high clamp value is reached or exceeded. The bit will automatically clear when the channel command value \(^1\) drops below the high clamp value. |

5.3.4 Under-Range Flag Bits U0 to U3 (Word 1)
Under-range bits for channels 0 through 3 are contained in word 1. They apply to all output types. When set (1), the under-range flag bit indicates an output signal that is at the minimum of its normal operating range for the represented channel or sensor. The module automatically resets (0) when the under-range condition is cleared and the data value is within the normal operating range.

| NOTE | This bit can also be set if the low clamp value is reached or exceeded. The bit will automatically clear when the channel command value \(^1\) is above the low clamp value. Refer to section 5.3.16 for more details. |

5.3.5 Hold Last State Bits H0 to H3 (Word 1)
The hold last state bit will be set on the associated channel if the channel is configured to output the last commanded value, and a fault or “run to program” condition is present.

5.3.6 Channel X\(^2\) Command Value Echo (Words 2 to 5)
Data words 2 through 5 correspond to channels 0 through 3 respectively and contain the current command value \(^1\) stored in the module’s RAM memory for the associated channel. These input words can be used as a handshake to verify that the module has received the correct command value for the associated channel. The command value is defined in section 5.4.2.

---

\(^1\) Module output words 0 through 3 for channel 0 through 3, respectively.
\(^2\) X refers to channel number 0 through 3
5.3.7 HART Packet Data (Words 6 to 25)
This block of twenty words contains the multiplexed HART packet data for all four channels.¹

5.3.8 Message Slave Control (Word 26)
The message slave control word controls how data is returned from the module after sending a message using output words 6 through 27².

5.3.9 Message Reply Size (Word 27)
The message response size indicates the number of bytes returned by the module after sending a message using output words 6 through 27².

5.3.10 Message Reply Buffer (Words 28…47)
After sending a message to the module, the response data for the message is stored in the message response buffer².

5.3.11 Reserved (Words 48…69)
Reserved for future expansion.

Module Configuration

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

Each channel defaults to:
- Channel disabled.
- Range type: 0 to 20 mA.
- Data Format: Raw Proportional.
- HART Communication disabled.

¹ For more details, refer to Chapter 6
² For more details, refer to Chapter 6
See associated dialog below:

The structure of the channel configuration file is shown below.

### Table 5-1. Module Configuration

<table>
<thead>
<tr>
<th>Word</th>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EC</td>
<td>Reserved</td>
<td>SV3</td>
<td>SV2</td>
<td>SV1</td>
<td>SV0</td>
<td>SIU</td>
<td>SIO</td>
<td>LA</td>
<td>ER</td>
<td>FM</td>
<td>PM</td>
<td>HI</td>
<td>PFE</td>
<td></td>
<td>Ch0 General Settings</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td>Data Format</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ch0 Output type and Data Format</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Channel 0 Fault Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ch0 Fault Value</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Channel 0 Program Idle Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ch0 Program Idle Value</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Channel 0 Low Clamp Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ch0 Low Clamp Value</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Channel 0 High Clamp Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ch0 High Clamp Value</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Channel 0 Ramp Rate Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ch0 Ramp Rate Value</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not Used (Set to Zero)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>EC</td>
<td>Reserved</td>
<td>SV3</td>
<td>SV2</td>
<td>SV1</td>
<td>SV0</td>
<td>SIU</td>
<td>SIO</td>
<td>LA</td>
<td>ER</td>
<td>FM</td>
<td>PM</td>
<td>HI</td>
<td>PFE</td>
<td></td>
<td>Ch1 General Settings</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td>Data Format</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ch1 Output type and Data Format</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Channel 1 Fault Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ch1 Fault Value</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Channel 1 Program Idle Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ch1 Program Idle Value</td>
<td></td>
</tr>
<tr>
<td>Word</td>
<td>Bit</td>
<td>Function</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>---------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>Channel 1 Low Clamp Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>Channel 1 High Clamp Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>Channel 1 Ramp Rate Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>EC</td>
<td>Ch1 General Settings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>Data Format</td>
<td>Reserved</td>
<td>Output Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>Channel 2 Fault Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>10</td>
<td>Channel 2 Program Idle Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>9</td>
<td>Channel 2 Low Clamp Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>8</td>
<td>Channel 2 High Clamp Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>7</td>
<td>Channel 2 Ramp Rate Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>6</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>EC</td>
<td>Ch2 General Settings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>Data Format</td>
<td>Reserved</td>
<td>Output Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>5</td>
<td>Channel 3 Fault Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>4</td>
<td>Channel 3 Program Idle Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>3</td>
<td>Channel 3 Low Clamp Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>2</td>
<td>Channel 3 High Clamp Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>Channel 3 Ramp Rate Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Set to Zero</td>
<td>Pad (16 Bit Alignment)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Reserved</td>
<td>PA</td>
<td>EH3</td>
<td>EH2</td>
<td>EH1</td>
<td>EH0</td>
<td>Handle Timeout</td>
<td>HART Configuration Word</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The table shows the bit assignments and functions for the module data, status, and channel configuration.
<table>
<thead>
<tr>
<th>Word</th>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td></td>
<td>Channel 0 HART Slot Variables 0 &amp; 1</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>Channel 0 HART Slot Variables 2 &amp; 3</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td>Channel 1 HART Slot Variables 0 &amp; 1</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>Channel 1 HART Slot Variables 2 &amp; 3</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td>Channel 2 HART Slot Variables 0 &amp; 1</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td>Channel 2 HART Slot Variables 2 &amp; 3</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>Channel 3 HART Slot Variables 0 &amp; 1</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td>Channel 3 HART Slot Variables 2 &amp; 3</td>
</tr>
</tbody>
</table>
### 5.3.12 Channel X General Settings (Words 0, 8, 16, 24)

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

**Table 5-2. Channel General Settings**

<table>
<thead>
<tr>
<th>To Select</th>
<th>Make these bit settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFE (Program/Idle to Fault Enable)</td>
<td></td>
</tr>
<tr>
<td>Disble</td>
<td>0</td>
</tr>
<tr>
<td>Enable</td>
<td>1</td>
</tr>
<tr>
<td>HI (Hold for Initialization)</td>
<td></td>
</tr>
<tr>
<td>Disble</td>
<td>0</td>
</tr>
<tr>
<td>Enable</td>
<td>1</td>
</tr>
<tr>
<td>PM (Program Mode)</td>
<td></td>
</tr>
<tr>
<td>Hold Last State</td>
<td>0</td>
</tr>
<tr>
<td>User Defined</td>
<td>1</td>
</tr>
<tr>
<td>FM (Fault Mode)</td>
<td></td>
</tr>
<tr>
<td>Hold Last State</td>
<td>0</td>
</tr>
<tr>
<td>User Defined</td>
<td>1</td>
</tr>
<tr>
<td>ER (Enable Ramping)</td>
<td></td>
</tr>
<tr>
<td>Disble</td>
<td>0</td>
</tr>
<tr>
<td>Enable</td>
<td>1</td>
</tr>
<tr>
<td>LA (Latch Alarms)</td>
<td></td>
</tr>
<tr>
<td>Disble</td>
<td>0</td>
</tr>
<tr>
<td>Enable</td>
<td>1</td>
</tr>
<tr>
<td>SIO (System Interrupt High Clamp, Over Range Alarm)</td>
<td></td>
</tr>
<tr>
<td>Disble</td>
<td>0</td>
</tr>
<tr>
<td>Enable</td>
<td>1</td>
</tr>
<tr>
<td>SIU (System Interrupt Low Clamp, Under Range Alarm)</td>
<td></td>
</tr>
<tr>
<td>Disble</td>
<td>0</td>
</tr>
<tr>
<td>Enable</td>
<td>1</td>
</tr>
<tr>
<td>Enable Slot Variable 0</td>
<td></td>
</tr>
<tr>
<td>Disble</td>
<td>0</td>
</tr>
<tr>
<td>Enable</td>
<td>1</td>
</tr>
<tr>
<td>Enable Slot Variable 1</td>
<td></td>
</tr>
<tr>
<td>Disble</td>
<td>0</td>
</tr>
<tr>
<td>Enable</td>
<td>1</td>
</tr>
<tr>
<td>Enable Slot Variable 2</td>
<td></td>
</tr>
<tr>
<td>Disble</td>
<td>0</td>
</tr>
<tr>
<td>Enable</td>
<td>1</td>
</tr>
<tr>
<td>Enable Slot Variable 3</td>
<td></td>
</tr>
<tr>
<td>Disble</td>
<td>0</td>
</tr>
<tr>
<td>Enable</td>
<td>1</td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>Set to Zero</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

1 Where X is the channel number
### Real Time Sample Value (Word 0)

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

### PFE (Program/Idle to Fault Enable)

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

![NOTE]

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

### HI (Hold for Initialization)

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

If Hold for Initialization is selected, outputs will hold if any of the three conditions occur:
- Initial connection is established after power-up.
- Communications re-established after a communications fault.
- There is a transition from Program mode to run mode.

### PM (Program Mode)

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

---

1 Entering a value of zero allows the module to automatically select the fastest allowed RTS rate.
NOTE

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

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

NOTE

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

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

LA (Latch Alarm)
The latch alarm feature enables process alarms (i.e. Over Range, Under Range, Clamp High, etc.) to be latched during an alarm condition.

NOTE

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

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

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

NOTE

Not all controllers support this function. Refer to your controller’s user manual for more details.
**SV0...SV3 (Slot Variables 0 to 3)**
Enables HART slot variables 0 through 3 for the selected channel.

| NOTE | Slot variables are not supported by all HART devices. |

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

**Reserved**
Must be set to Zero

**EC (Enable Channel)**
This bit allows the user to enable or disable the channel.

**5.3.13 Channel X’ Output Type and Data Format (Words 1, 9, 17, 25)**
This section of the configuration allows the user to define the output type (i.e. 0 to 20 mA or 4 to 20 mA) and the data format for the associated channel. See table below.

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

**Table 5-3 Output Type and Data Format**

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

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

**Reserved**

Reserved for future expansion and should be set to zero.

**Data Format**

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

- **Raw/Proportional Data**
  
  The raw/proportional data format provides the best resolution of all the supported data formats.
  
  If you select the raw/proportional data format for a channel, the output command word\(^1\) will be a number between -32767 and +32767. For example, if a 4 to 20 mA output type is selected, 4 mA corresponds to -32767 counts and 20 mA corresponds to +32767.

- **Engineering Units**
  
  When using this data format, the module scales the output command word\(^1\) to the actual engineering values for the selected output type. Values are expressed with an assumed decimal place. Refer to Table 5-4 (Data Formats).
  
  The resolution for the selected data format is dependent on the output type selected.

- **Scaled-for-PID**
  
  The PID algorithm for the MicroLogix 1500 and other Allen-Bradley controllers (i.e. SLC) utilize an engineering range of 0 to 16383. The scaled for PID format allows the programmer to assign the output command word\(^1\) for the selected channel directly to the PID instruction’s CV variable, thus eliminating the need for further scaling.

- **Percent Range**
  
  The output command word\(^1\) for the selected channel is scaled using a range of 0 to +10000 counts. For example, if a 4 to 20 mA output type is selected, 4 mA corresponds to 0 counts and 20 mA corresponds to +10000 counts.

---

\(^1\) Module output words 0 through 3 for channel 0 through 3, respectively.
NOTE

10000 counts represent 100.00 percent.

Table 5-4 Data Formats

<table>
<thead>
<tr>
<th>Output Type</th>
<th>Engineering</th>
<th>Scaled for PID</th>
<th>Proportional Counts</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 20 mA</td>
<td>0 to 20000</td>
<td>0 to 16383</td>
<td>-32768 to 29646</td>
<td>0 to 10000</td>
</tr>
<tr>
<td>4 – 20 mA</td>
<td>4000 to 20000</td>
<td>0 to 16383</td>
<td>-29822 to 29085</td>
<td>0 to 10000</td>
</tr>
</tbody>
</table>

Reserved
Reserved for future expansion and should be set to zero.

5.3.14 Fault Value (Words 2, 10, 18, 26)
During a processor fault condition, the associated output channel will be driven to this value.

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

5.3.15 Program/Idle Value (Words 3, 11, 19, 27)
During a processor program or idle mode, the associated output channel will be driven to this value.

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

5.3.16 Low Clamp Value (Words 4, 12, 20, 28)
The low clamp value determines the minimum analog signal that will be applied to the associated channel.

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

\(^1\) Module output words 0 through 3 for channel 0 through 3, respectively.
The low clamp value must be less than the high clamp value or a module fault will be generated. If both the low clamp and high clamp values are left at zero, the output is clamped using the end points of the associated channel’s defined output range.

### 5.3.17 High Clamp Value (Words 5, 13, 21, 29)

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

**NOTE**

If the channel command value\(^1\) is greater than or equal to the high clamp value, the over range status bit, for the associated channel, will be set.

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

### 5.3.18 Ramp Rate Value (Words 6, 14, 22, 30)

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

<table>
<thead>
<tr>
<th>Output Data Format Output Range/Type</th>
<th>Total Counts in Full Scale</th>
<th>Number of Counts for Every 1% of Ramp Rate</th>
<th>Real Units/Second for Every 1% of Ramp Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proportional Counts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 to 20 mA</td>
<td>65534</td>
<td>655</td>
<td>0.2 mA/Sec</td>
</tr>
<tr>
<td>4 to 20 mA</td>
<td>65534</td>
<td>655</td>
<td>0.16 mA/Sec</td>
</tr>
<tr>
<td>Engineering Units</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 to 20 mA</td>
<td>21000</td>
<td>210</td>
<td>0.2 mA/Sec</td>
</tr>
<tr>
<td>4 to 20 mA</td>
<td>17800</td>
<td>178</td>
<td>0.16 mA/Sec</td>
</tr>
<tr>
<td>Scaled for PID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 to 20 mA</td>
<td>16383</td>
<td>164</td>
<td>0.2 mA/Sec</td>
</tr>
<tr>
<td>4 to 20 mA</td>
<td>16383</td>
<td>164</td>
<td>0.16 mA/Sec</td>
</tr>
<tr>
<td>Percent of Full Scale</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 to 20 mA</td>
<td>10000</td>
<td>100</td>
<td>0.2 mA/Sec</td>
</tr>
</tbody>
</table>
5.3.19 Spare (Words 7, 15, 23, 31)
Reserved for future expansion

5.3.20 Pad (Word 32)
1-bit alignment (Set to Zero)

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

<table>
<thead>
<tr>
<th>Output Data Format Output Range/Type</th>
<th>Total Counts in Full Scale</th>
<th>Number of Counts for Every 1% of Ramp Rate</th>
<th>Real Units/Second for Every 1% of Ramp Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 to 20 mA</td>
<td>10000</td>
<td>100</td>
<td>0.16 mA/Sec</td>
</tr>
</tbody>
</table>

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

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

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

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

**NOTE**

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

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

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

## Pass-Through Scheme

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

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

## NOTE

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

### Enable HART Logging

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

5.3.22 Channel X\(^1\) HART Slot Variables 0 & 1 (Words 34, 36, 38, 40)

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

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

5.3.23 Channel X\(^1\) HART Slot Variables 2 & 3 (Words 35, 37, 39, 41)

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

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

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

### Section 5.4

**Output Data File**

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

---

\(^1\) Where X is the channel number (0 to 3)
### Table 5-6 Output Data File

<table>
<thead>
<tr>
<th>Word/Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Channel 0 Command Value</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Channel 1 Command Value</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Channel 2 Command Value</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Channel 3 Command Value</td>
</tr>
<tr>
<td>4</td>
<td>HS3</td>
<td>HS2</td>
<td>HS1</td>
<td>HS0</td>
<td></td>
<td></td>
<td>UL3</td>
<td>UH3</td>
<td>UL2</td>
<td>UH2</td>
<td>UL1</td>
<td>UH1</td>
<td>UL0</td>
<td>UH0</td>
<td></td>
<td>HS3 through HS0 are used to set the high status values for channels 0 through 3. The reserved bits are not used.</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Packet Just Scanned</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Message Master Control</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Message Request Size</td>
</tr>
<tr>
<td>8..27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Message Request Buffer</td>
</tr>
<tr>
<td>28..49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### 5.4.2 Channel X Command Value

The command value is used to control the analog output signal for each channel. The command value is entered in engineering units defined by the channel’s selected data format and output type. Refer to section 5.3.13 for more details regarding data format and output type.

### 5.4.3 Unlatch Process High Alarms UH0 to UH3 (Word 4)

UH0 through UH3 will unlatch the high process alarms for channels 0 through 3 respectively. Refer to section 0 for more information regarding setting the alarm latch function. To unlatch the high process alarm on a given channel, set the unlatch bit to 1.

#### NOTE

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

#### NOTE

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

#### NOTE

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

### 5.4.4 Unlatch Process Low Alarms UL0 to UL3 (Word 4)

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

**NOTE**

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

**NOTE**

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

**NOTE**

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

### 5.4.5 Hart Suspend HS0 to HS3 (Word 4)

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

**NOTE**

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

### 5.4.6 Packet Just Scanned (Word 5)

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

**NOTE**

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

---

1 Refer to Chapter 7 for more details.
5.4.7 Message Master Control (Word 6)
This word is used to control the data flow of a message sent to the module. These messages include module commands such as HART pass-through, HART suspend and resume, and get device information.¹

5.4.8 Message Request Size (Word 7)
The *message request size* determines the size of the message, in bytes, that will be sent to the module.¹

5.4.9 Message Request Buffer (Words 8…27)
The *message request buffer* contains the data making up the message that will be sent to the module.¹

5.4.10 Reserved (Words 28…49)
Reserved for future expansion.

### Section 5.5 Determining Module Update Time

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

#### 5.5.1 Module Update Time

<table>
<thead>
<tr>
<th></th>
<th>HART Enabled</th>
<th>HART Disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>12 ms</strong></td>
<td>10 ms</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 6
Enabling and Using HART on the 1769sc-OF4IH

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

The chapter is broken down into the following sections:

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

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

Section 6.1
Configuring the Module for HART

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

For HART to be active on any given channel, the channel configuration must contain the following basic settings.
The channel must be enabled, set for 4 to 20 mA, and the Enable HART Communication checkbox must be checked.

**NOTE**

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

### Section 6.2
**HART Packet Data**

#### 6.2.1 How the Module Connects to a Field Device

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

HART multi-drop is not supported by the OF4IH.

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

Module specific commands include the HART pass-through commands, HART suspend, HART resume, and the get HART device information command. Gathering HART data is accomplished using two processes auto acquisition, and or using the module specific commands.

### 6.2.2 Auto Acquisition

When a channel is configured for HART, the module will automatically search and establish a connection to any HART field device wired to the channel. Once the module establishes a connection it will begin to acquire HART data, including device specific codes (i.e. Manufacturer ID, serial number, etc.), the four dynamic variables, extended device status, slot variables (if enabled), and any stored ASCII message descriptor that may be present. The HART data retrieved automatically by the module is then displayed in the input image (Of4ih0Input.HartData) and is accessible by ladder logic. The HART data will update, on average, every 3.5 seconds if all four channels are enabled for HART. The module initiates the connection by sending a string of HART commands to the field device. Please see figure below.
The data that is collected from the process described in Figure 7-4 (Auto Acquisition Flow) is buffered to the module RAM memory. Since the amount of data returned from the auto-acquisition process is extensive, the data is multiplexed into five separate packets and for each individual channel. The multiplexed data can be read from a 40-byte array that is located in the OF4IH0Input.HartData tag. The multiplexed data is demultiplexed using ladder and stored in five different arrays which are structured using packets zero through four. The packets are defined as “user defined data types” and can be seen in Table 6-1 through Table 6-5.
Table 6-1. HART Packet 0

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

\(^{17} X \text{ represents the module channel number (0 to 3)}\)
### Table 6-2. HART Packet 1

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

---

18 X represents the module channel number (0 to 3)
### Chapter 6: Enabling and Using HART on the 1769sc-OF4IH

#### Table 6-3. HART Packet 2

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

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

---

19 X represents the module channel number (0 to 3)
### Table 6-4. HART Packet 3

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

### Table 6-5. HART Packet 4

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Data Type</th>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>If4ih0Packet4</td>
<td>Packet4[4,1]</td>
<td>NA</td>
<td>Two-dimensional array containing packet 4 data for all 4 channels.</td>
</tr>
<tr>
<td>If4ih0Packet4[X,0].HartChannelID</td>
<td>INT</td>
<td>BIN</td>
<td>Bits 0 to 3: Channel number (0 – 3). Bit 4: Searching/Initializing HART device Bit 5: HART communication failure or device not found Bit 6: Pass-through message pending (ready) Bit 7: Unused (0) Bits 8 to 10: Packet ID Bit 11 through 15: Unused</td>
</tr>
<tr>
<td>If4ih0Packet4[X,0].Date</td>
<td>SINT[3]</td>
<td>DEC</td>
<td>Stored date in the field device</td>
</tr>
</tbody>
</table>
Chapter 6: Enabling and Using HART on the 1769sc-OF4IH

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Data Type</th>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>If4ih0Packet4[X,0].FinalAssemblyNumber</td>
<td>SINT[3]</td>
<td>DEC</td>
<td>The final assembly number is used for identifying the materials and electronics that comprise the field device.</td>
</tr>
<tr>
<td>If4ih0Packet4[X,0].ExtendedStatus</td>
<td>SINT[24]</td>
<td>DEC</td>
<td>The extended status returned by HART command 48</td>
</tr>
<tr>
<td>If4ih0Packet4[X,0].Pad</td>
<td>SINT[3]</td>
<td>DEC</td>
<td>Pad 32-bit alignment</td>
</tr>
</tbody>
</table>

**NOTE**

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

The ladder determines which packet to copy the data to, by monitoring the state of bits 0, 1, 2 and 8, 9, 10, found in the first two bytes of the OF4IH0Input.HartData tag. Bits 0, 1, 2 determine the current channel being scanned and bits 8, 9, and 10 determine the packet number. The ladder example, shown in the figure, performs this operation.
Figure 6-3. Packet Ladder

The following rung reads the current channel number being scanned and stores the value in the Of4ih0ChannelNumber tag.

```
  MVM
  Masked Move
  Source  Of4ih0Input.HartData[0]
  Mask  3
  Dest  Of4ih0ChannelNumber
```

The following rung reads the HART packet number and stores the value in the Of4ih0PacketNumber tag.

```
  MVM
  Masked Move
  Source  Of4ih0Input.HartData[1]
  Mask  7
  Dest  Of4ih0PacketNumber
```

The Of4ih0Packet0 is a two dimensional array. The first dimension is the channel number and the second dimension is the packet structure which is defined by the Packet0 user defined data type. When Of4ih0PacketNumber is equal to 0, the data from Of4ih0Input.HartData is copied to Of4ih0Packet0 for the appropriate channel depending on the current value stored in the Of4ih0ChannelNumber.

```
  EQU
  Source A  Of4ih0PacketNumber
  Source B  0

  COP
  Copy File
  Source  Of4ih0Input.HartData[0]
  Dest  Of4ih0Packet0[Of4ih0ChannelNumber,0]
  Length  1
```

The Of4ih0Packet1 is a two dimensional array. The first dimension is the channel number and the second dimension is the packet structure which is defined by the Packet1 user defined data type. When Of4ih0PacketNumber is equal to 1, the data from Of4ih0Input.HartData is copied to Of4ih0Packet1 for the appropriate channel depending on the current value stored in the Of4ih0ChannelNumber.

```
  EQU
  Source A  Of4ih0PacketNumber
  Source B  1

  COP
  Copy File
  Source  Of4ih0Input.HartData[0]
  Dest  Of4ih0Packet1[Of4ih0ChannelNumber,0]
  Length  1
```

The Of4ih0Packet2 is a two dimensional array. The first dimension is the channel number and the second dimension is the packet structure which is defined by the Packet2 user defined data type. When Of4ih0PacketNumber is equal to 2, the data from Of4ih0Input.HartData is copied to Of4ih0Packet2 for the appropriate channel depending on the current value stored in the Of4ih0ChannelNumber.

```
  EQU
  Source A  Of4ih0PacketNumber
  Source B  2

  COP
  Copy File
  Source  Of4ih0Input.HartData[0]
  Dest  Of4ih0Packet2[Of4ih0ChannelNumber,0]
  Length  1
```
The ladder in Figure 6-4 can be found in the project sample file located on our website at www.spectrumcontrols.com

### 6.2.3 Packet Interval

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

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

The amount of time saved using this method depends on the scan time of the ladder and the update time of each individual HART transmitter.
Section 6.3
Sending and Receiving Messages

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

6.3.1 Module Output Tags Used for Messaging
The OF4IH module uses 28 words for sending messages and controlling data flow. The table below shows the output image for the OF4IH module. For more detail regarding module output image, refer to Section 5.4.

| Word/Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 1        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 2        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 3        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 4        | HS3| HS2| HS1| HS0|     |    |    |    |    |    |    |    |    |    |    |
| 5        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 6        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 7        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 8..27    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 28..49   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Word 6 (Message Master Control)
The message master control initiates the paging process and controls the flow of data to and from the module. The data flow control is accomplished by using the message master control with the message slave control to manage which pages are being sent and what direction the page is going, that is, whether the page is being sent to the module or read from the module.
### Word 7 (Message Request Size)

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

### Words 8…27 (Message Request Buffer)

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

### 6.3.2 Module Input Tags Used For Messaging

The module uses 48 input words to receive messages and control data flow. The table below shows the input words used by the module. Refer to Chapter 6 for more information regarding input words 0 through 48.

---

#### Table: Module Input Tags Used For Messaging

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

---

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

<table>
<thead>
<tr>
<th>RR</th>
<th>SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page being sent (Page = 38 Bytes)</td>
<td></td>
</tr>
<tr>
<td>Page last received</td>
<td></td>
</tr>
</tbody>
</table>
### Chapter 6: Enabling and Using HART on the 1769sc-OF4IH

#### Table 6-7. Module Input Table

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

**Word 26 (Message Slave Control)**

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

#### Table 6-8. Paging Error Codes

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

**Word 27 (Message Response Size)**

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

**Words 28...47 (Message Response Buffer)**

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

---

<sup>20</sup> Changing bit values is not supported by all controllers. Refer to your controller manual for details.
6.3.3 Processing a Message

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

1. Store the message you wish to send to the module in an array. Remember the message can be up to 257 bytes long, so make the array large enough.

2. Copy the first page of data, up to 38 bytes, to the message request buffer. If the number of bytes is odd, the last byte in the last word will be padded with a zero.

3. Calculate the checksum of the message by taking the exclusive OR of all the words within the page (19 max). Place the result into the last word of the message (i.e. word # 20 if a full page).

4. Enter the size of the message to be sent to the module into the message request size output word.

5. Add a 1 to the lower nibble of the message master control word (i.e. 0001Hex). The message master control should be zero when the message is started.

9. Wait for the module to reply that it has received the page without error, by monitoring the second nibble of the message slave control (i.e. 0100). If the lower nibble contains FF, stop the process because the data is corrupted. The first byte in the message response buffer will contain the paging error code. Refer to Table 6-1 for a description of the errors.

10. Check to see if there are more pages to send by comparing the bytes sent to the message request size. If so, repeat steps 2 through 6. If not, go to step 8.

11. Monitor the lower nibble of the message slave control to see if the first page of the response data is ready (0101).

12. Copy the first page of the response data from the message response buffer to a temporary array.

13. Take the exclusive OR of all the words within the page (19 max) with the exception of the last word which is the checksum. Compare the calculated checksum with the checksum stored in the last byte. If they are equal, go to step 11. If they are not, stop the process because the data is corrupted.

14. Check to see if there is more response data remaining by comparing the bytes received to the message response size. If so, repeat steps 8 through 10. If not, the message is finished. To send another message clear the message master control and repeat the process.

A graphical representation of the process can be seen in Figure 6-2 and 6-3.
Figure 6-7. Sending Message

- **MsgMasterControl (Hex)** = RR|SS
  - RR = Page Last Received
  - SS = Page Being Sent

- **MsgRequestSize** = Total size of message in bytes, up to 257 bytes.

- **MsgRequestBuffer** = One page of data being sent to module. Last byte is page checksum. 1 page = 38 bytes max.

First Page
- 38 Bytes
- **MsgMasterControl = 00|01**
- **MsgSlaveControl = 00|00**
- Bytes sent ≠ MsgRequestSize

Second Page
- 38 Bytes
- **MsgMasterControl = 00|02**
- **MsgSlaveControl = 01|00**
- Bytes sent ≠ MsgRequestSize

nth Page
- 38 Bytes
- **MsgMasterControl = 00|01**
- **MsgSlaveControl = 00|00**
- Bytes sent = MsgRequestSize

If checksum is valid, then ready to receive data from module
Chapter 6: Enabling and Using HART on the 1769sc-OF4Ih

Figure 6-8. Receiving Message

Message Returned

MsgMasterControl = 00|02
MsgSlaveControl = 02|01
Bytes received <>

MsgSlaveControl (Hex) = RR|SS
RR = Page Last Received
SS = Page Being Sent

MsgResponseSize = Total size of response message, up to 257 bytes.

MsgResponseBuffer = One page of data being sent to the PLC. Last byte is page checksum. 1 page = 38 bytes max.

First Page
Up to 257 Bytes

Second Page

nth Page

Message Returned

MsgMasterControl = 01|02
MsgSlaveControl = 02|02
Bytes received <> MsgResponseSize

MsgSlaveControl (Hex) = RR|SS
RR = Page Last Received
SS = Page Being Sent

MsgResponseSize = Total size of response message, up to 257 bytes.

MsgResponseBuffer = One page of data being sent to the PLC. Last byte is page checksum. 1 page = 38 bytes max.

First Page
Up to 257 Bytes

Second Page

nth Page

Message Returned

MsgMasterControl = 02|02
MsgSlaveControl = 02|02
Bytes received = MsgResponseSize

MsgSlaveControl (Hex) = RR|SS
RR = Page Last Received
SS = Page Being Sent

MsgResponseSize = Total size of response message, up to 257 bytes.

MsgResponseBuffer = One page of data being sent to the PLC. Last byte is page checksum. 1 page = 38 bytes max.

First Page
Up to 257 Bytes

Second Page

nth Page

If checksum is valid, then message complete

MsgResponseSize = Total size of response message, up to 257 bytes.

MsgResponseBuffer = One page of data being sent to the PLC. Last byte is page checksum. 1 page = 38 bytes max.
Figure 6-9a. Message Ladder

The following rung resets the message paging logic.

0

Subroutine
Input Par Of4ih0.Msg.Reset
Input Par Of4ih0.Msg.SourceSize
Input Par Of4ih0.Msg.SourceData

The following rung initializes the message paging logic within the module.

1

Clear
Dest Of4ih0.Msg.Step 0

FILL File
Source Dest Of4ih0.Msg.DestinationData[0] Length 256

Clear
Dest Of4ih0.Msg.Reset (U)

The following rung calculates the bytes remaining for the current message being sent.

2

Equal
Source A 0
Source B 0

Equal
Source A 0
Source B 1

Clear
Dest Of4ih0.Msg.SourceIndex 1

Clear
Dest Of4ih0.Msg.SourceBytesSent 3

Move
Source 1
Dest Of4ih0.Msg.SourceIndex 1

Move
Source 3
Dest Of4ih0.Msg.RequestSize 3

Move
Source 3
Dest Of4ih0.Msg.Step 0

Subtract
Source A 0
Source B 0
Dest Of4ih0.Msg.SourceBytesRemaining 3
Figure 6-9b.

If the bytes remaining are less than 36, this rung copies the data for the current page being sent to the message request buffer. This rung also calculates the data index.

```plaintext
4

Equal
Source A 04h0Msg Step 0
Source B 1

Not Equal
Source A 04h0Msg SourceBytesRemaining
Source B 0

Less Than (A=B)
Source A 04h0Msg SourceBytesRemaining
Source B 36

Copy File
Source 04h0Msg SourceData[04h0Msg SourceDataIndex]
Dest 04h0Output.MsgRequestBuffer[0]
Length 04h0Msg SourceBytesRemaining

Add
Source A 04h0Msg SourceBytesRemaining
Source B 04h0Msg SourceBytesSent
Dest 04h0Msg SourceBytesSent

Move
Source 04h0Msg SourceBytesRemaining
Dest 04h0Msg SourceTempIndex

Move
Source 04h0Msg SourceBytesSent
Dest 04h0Msg SourceDataIndex
```

User's Manual 0300217-04 Rev. A
Figure 6-9c.

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

The following rung determines whether an odd number of bytes have been copied. If so, the last byte in the last word is padded with a zero.
This rung copies the source data for the page to a temporary array, so that the page checksum can be calculated.

Figure 6-9d.

This rung calculates the checksum for the page of data being sent to the module.

User's Manual 0300217-04 Rev. A
Figure 6-9e.

This rung stores the checksum in the last word for the page.

---

Source XOR Control

AND

Source A: 04h00_msg SourceTempIndex
Source B: 1
Dest: Temp1
256

EQU

Source A: Temp1
Source B: 04h00_msg SourceTempIndex
256

ADD

Source A: 04h00_msg SourceTempIndex
Source B: 1
Dest: 04h00_msg SourceTempIndex
255

COP

Copy File
Source: 04h00_output_msg RequestBuffer[04h00_msg SourceTempIndex]
Dest: 04h00_msg SourceCheckSum
Length

MVM

Masked Move
Source: 04h00_msg SourcePageIndex
Mask: 255
Dest: 04h00_output_msg MasterControl
16#0101

MOV

Source: 2
Dest: 04h00_msg Step
0

Resets checksum logic

Source XOR Control

CLR

Clear
Dest: 04h00_msg SourceCheckSum
0

Check for message corruption

EQU

Source A: 04h00_msg Step
Source B: 2

AND

Source A: 04h00_input_msg SlaveControl
16#0101
Source B: 255
Dest: Temp1
256

EQU

Source A: Temp1
Source B: 255

04h00_msg ER

EndNow
JMP
Figure 6-9f.

Wait for module to echo back page received.

More data to send?

If done sending data, start reading response.
Figure 6-9g.

Check for message corruption

15
EQU
Equal
Source A 0x40hMsg Step 0
Source B 4

AND
Bitwise AND
Source A 0x40hInput.MsgSlaveControl 16#0101
Source B 255
Dest Temp1 256

EQU
Equal
Source A Temp1 256
Source B 255

Wait for next response page.

16
EQU
Equal
Source A 0x40hMsg Step 0
Source B 4

AND
Bitwise AND
Source A 0x40hInput.MsgSlaveControl 16#0101
Source B 255
Dest Temp1 256

EQU
Equal
Source A 0x40hMsg DestPageIndex 1
Source B Temp1 256

MOV
Move
Source 5
Dest 0x40hMsg Step 0

The following rung calculates the bytes remaining for the current message being received.

17
EQU
Equal
Source A 0x40hMsg Step 0
Source B 5

SUB
Subtract
Source A 0x40hInput.MsgResponseSize 18
Source B 0x40hMsg.DestBytesSent 16
Dest 0x40hMsg.DestBytesRemaining 16

The following rung sets the position of the pointer used to calculate the checksum for the current page of data being received.

18
EQU
Equal
Source A 0x40hMsg Step 0
Source B 5

MOV
Move
Source 0x40hMsg.DestDataIndex 18
Dest Temp2 0
Figure 6-9h.

If the bytes remaining are less than 38, this rung copies the data from the message response buffer to a temporary array were the data can be viewed. This rung also calculates the data index.

- **EOU**
  - Equal
    - Source A: 04hh0Msg.SendStep
    - Source B: 5
  - Not Equal
    - Source A: 04hh0Msg.DestBytesRemaining
    - Source B: 16

- **LEO**
  - Less Than (A<B)
    - Source A: 04hh0Msg.DestBytesRemaining
    - Source B: 38

- **NEO**
  - Source A: 04hh0Msg.DestBytesRemaining
  - Source B: 16

- **COP**
  - Copy File
    - Source: 04hh0Input.MsgResponseBuffer[0]
    - Dest: 04hh0Msg.DestData[04hh0Msg.DestDataIndex]
    - Length: 04hh0Msg.DestBytesRemaining

- **ADD**
  - Add
    - Source A: 04hh0Msg.DestBytesRemaining
    - Source B: 16
    - Dest: 04hh0Msg.DestBytesSent

- **MOV**
  - Move
    - Source: 04hh0Msg.DestBytesSent
    - Dest: 04hh0Msg.DestTempIndex

- **MOV**
  - Move
    - Source: 04hh0Msg.DestBytesSent
    - Dest: 04hh0Msg.DestDataIndex
Figure 6-9i.

If the bytes remaining are greater than or equal to 38, this rung copies the data from the message response buffer to a temporary array were the data can be viewed. This rung also calculates the data index.

The following rung determines whether an odd number of bytes have been copied. If so, the last byte in the last word is padded with a zero.
Figure 6-9j.

This rung copies the source data for the page to a temporary array, so that the page checksum can be calculated.

This rung calculates the checksum for the current page of data being received from the module.

```
Figure 6-9j.

This rung copies the source data for the page to a temporary array, so that the page checksum can be calculated:

```

```

This rung calculates the checksum for the current page of data being received from the module:

```

```

User's Manual 0300217-04 Rev. A
Figure 6-9k.

This rung checks for message corruption by comparing the calculated checksum with the checksum stored in the message response buffer.

DestXORControl.DN

- Equal
  - Source A: Temp1
  - Source B: 255

- ADD
  - Source A: O41h0Msg.DestTempIndex
  - Source B: 1
  - Dest: Temp1
  - Value: 256

- COP
  - Source: O41h0Input.MsgResponseBuffer[O41h0Msg.DestTempIndex]
  - Dest: Length

- NEO
  - Source A: O41h0Msg.DestChecksum
  - Source B: 0
  - Temp3: -4011

- MUL
  - Source A: O41h0Msg.DestPageIndex
  - Source B: 256
  - Dest: Temp1
  - Value: 256

- MVM
  - Source: Temp1
  - Mask: 65280
  - Dest: O41h0Output.MsgMasterControl
  - Value: 1940101

- MOV
  - Source: 6
  - Dest: O41h0Msg.Steps
  - Value: 0

- CLR
  - RES
  - Source: 0
  - Dest: O41h0Msg.DestChecksum
  - Value: 0

- Not Equal
  - Source A: O41h0Msg.Steps
  - Source B: 5

-RESET checksum logic

User's Manual 0300217-04 Rev. A
Section 6.4
Module Specific Commands

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

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

Figure 6-10. Message Flow

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

6.4.1 Get HART Device Information

The Get HART Device Information command is used to gather the device specific information for the connected HART device. The data that is retrieved can be seen in Table 6-10. The information that is gathered by this command is similar to the information gathered from the auto-acquisition process. The key difference is that the Get HART Device Information command pulls the data that has been stored in the module RAM and not directly from the field device.

Table 6-9. Get HART Device Information Command

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HART Channel Number</td>
<td>0x00 – 0x03 (1 byte)</td>
<td>Module input channel number for HART command</td>
</tr>
</tbody>
</table>
### HART Get Device Information – command message packet structure

Get currently cached Device Information for a given channel.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Number</td>
<td>0x03 (1 byte)</td>
<td>The command number to obtain HART device information</td>
</tr>
</tbody>
</table>

#### Table 6-90. Response If Device Information Is Not Available

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HART Channel Number</td>
<td>0x00 – 0x03 (1 byte)</td>
<td>Module input channel number for HART command</td>
</tr>
<tr>
<td>Status</td>
<td>(1 byte)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>34 = DR_RUNNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35 = DR_DEAD (bad request)</td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td>(1 byte)</td>
<td>Set to 1</td>
</tr>
<tr>
<td>Handle</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### HART Get Device Information - reply packet structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HART Channel Number</td>
<td>0x00 – 0x03 (1 byte)</td>
<td>Module input channel number for HART command</td>
</tr>
<tr>
<td>Status</td>
<td>00 = SUCCESS</td>
<td>Command status</td>
</tr>
<tr>
<td>Count</td>
<td>(1 byte)</td>
<td>Number of data bytes to following.</td>
</tr>
<tr>
<td>HART ManufacturerIDCode</td>
<td>(1 byte)</td>
<td>CMD#0, Byte 1</td>
</tr>
<tr>
<td>HARTDeviceTypeCode</td>
<td>(1 byte)</td>
<td>CMD#0, Byte 2</td>
</tr>
<tr>
<td>HARTPreamble</td>
<td>(1 byte)</td>
<td>CMD#0, Byte 3</td>
</tr>
<tr>
<td>HARTUnivCmdCode</td>
<td>(1 byte)</td>
<td>CMD#0, Byte 4</td>
</tr>
<tr>
<td>HARTTransSpecRev</td>
<td>(1 byte)</td>
<td>CMD#0, Byte 5</td>
</tr>
<tr>
<td>HARTSoftwareRevision</td>
<td>(1 byte)</td>
<td>CMD#0, Byte 6</td>
</tr>
<tr>
<td>HARTHardwareRevision</td>
<td>(1 byte)</td>
<td>CMD#0, Byte 7</td>
</tr>
<tr>
<td>HARTFlags</td>
<td>(1 byte)</td>
<td>CMD#0, Byte 8</td>
</tr>
<tr>
<td>Pad for 32-bit alignment</td>
<td>(1 byte)</td>
<td></td>
</tr>
</tbody>
</table>

#### Table 6-101. Response When Device Information Is Available
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARTDeviceIDNumber</td>
<td>(3 bytes) Device ID number</td>
<td>CMD#0, Bytes 9-11</td>
</tr>
<tr>
<td>Pad for 32-bit alignment</td>
<td>(1 byte)</td>
<td></td>
</tr>
<tr>
<td>HARTTag</td>
<td>(8 bytes unpacked ASCII)</td>
<td>CMD#13, Bytes 0-5</td>
</tr>
<tr>
<td>HARTDescriptor</td>
<td>(16 bytes unpacked ASCII)</td>
<td>CMD#13, Bytes 6-17</td>
</tr>
<tr>
<td>HARTDate</td>
<td>(3 bytes)</td>
<td>CMD#13, Bytes 18-20</td>
</tr>
<tr>
<td>Pad for 32-bit alignment</td>
<td>(1 byte)</td>
<td></td>
</tr>
<tr>
<td>HARTFinalAssemblyNumber</td>
<td>(3 bytes)</td>
<td>CMD#16, Bytes 0-2</td>
</tr>
<tr>
<td>Pad for 32-bit alignment</td>
<td>(1 byte)</td>
<td></td>
</tr>
<tr>
<td>HARTMessage</td>
<td>(32 bytes unpacked ASCII)</td>
<td>CMD#12, Bytes 0-23</td>
</tr>
<tr>
<td>HARTPVCode</td>
<td>(1 byte)</td>
<td>CMD#50, Bytes 0, 0xff if not supported</td>
</tr>
<tr>
<td>HARTSVCode</td>
<td>(1 byte)</td>
<td>CMD#50, Bytes 1, 0xff if not supported</td>
</tr>
<tr>
<td>HARTTVCode</td>
<td>(1 byte)</td>
<td>CMD#50, Bytes 2, 0xff if not supported</td>
</tr>
<tr>
<td>HARTQVCode</td>
<td>(1 byte)</td>
<td>CMD#50, Bytes 3, 0xff if not supported</td>
</tr>
<tr>
<td>HARTPVUnits</td>
<td>(1 byte)</td>
<td>CMD#3, Byte 4</td>
</tr>
<tr>
<td>HARTSVUnits</td>
<td>(1 byte)</td>
<td>CMD#3, Byte 9, 0 if not present</td>
</tr>
<tr>
<td>HARTTVUnits</td>
<td>(1 byte)</td>
<td>CMD#3, Byte 14, 0 if not present</td>
</tr>
<tr>
<td>HARTQVUnits</td>
<td>(1 byte)</td>
<td>CMD#3, Byte 19, 0 if not present</td>
</tr>
<tr>
<td>HARTSlot0Units</td>
<td>(1 byte)</td>
<td>CMD#33, Byte 1, 0 if not present Output module use only.</td>
</tr>
<tr>
<td>HARTSlot1Units</td>
<td>(1 byte)</td>
<td>CMD#33, Byte 7, 0 if not present Output module use only.</td>
</tr>
<tr>
<td>HARTSlot2Units</td>
<td>(1 byte)</td>
<td>CMD#33, Byte 13, 0 if not present</td>
</tr>
</tbody>
</table>
Chapter 6: Enabling and Using HART on the 1769sc-OF4IH

6-33

HART Get Device Information - reply packet structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARTSlot3Units</td>
<td>(1 byte)</td>
<td>CMD#33, Byte 19, 0 if not present Output module use only.</td>
</tr>
<tr>
<td>HARTPVLowerRange</td>
<td>(4 bytes – Floating Point Value)</td>
<td>CMD#15, Bytes 3-6</td>
</tr>
<tr>
<td>HARTPVUpperRange</td>
<td>(4 bytes – Floating Point Value)</td>
<td>CMD#15, Bytes 7-10</td>
</tr>
<tr>
<td>Pad for 32-bit alignment</td>
<td>(3 bytes)</td>
<td></td>
</tr>
</tbody>
</table>

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

SUCCESS will be sent back when all - the following conditions are met:
- Command and HART Channel number are both valid.
- HART channel device information is available.

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

DEAD will be sent back if any of the following conditions is true:
- Command or HART Channel number is invalid
- HART channel is not enabled
- HART communication has not been established, meaning that the 5-byte unique address has not been determined yet.
- All other conditioned not generating RUNNING or SUCCESS.

6.4.2 HART Pass-Through Command

The HART Pass-Through Command can be used to send any HART command including universal, common practice or device specific, directly to a field device. The module in this case could be considered a HART bridge. There can be two (2) instances of a HART pass-through message being serviced, meaning the pass-through message queue is 2 deep. The HART pass-through response will be queued the moment the command is received, if the queue spaces are not already in use, and be dispatched after at least a full scan is done. In other words, after servicing a pass-through, the HART module will make sure all enabled HART channels have updated variable values before another pass-through is placed into service.
All HART pass-through commands require a series of messages to be exchanged. First, a pass-through command request must be sent to the HART module to initiate the pass-through command. The HART module will respond to the command request with a command request reply that includes a handle that can be used to obtain the pass-through message response. Once the handle is received, the user may issue a Get Command Query to obtain the status of the pass-through command and the pass-through command response data, if it is available.

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

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

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

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

Table 6-11. HART Pass -Through Request Command

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HART Channel Number</td>
<td>0×00 – 0×03 (1 byte)</td>
<td>Module input channel number for HART command</td>
</tr>
<tr>
<td>Command Number</td>
<td>0×01 (1 byte)</td>
<td>The command number to issue a HART pass-through command.</td>
</tr>
</tbody>
</table>
Chapter 6: Enabling and Using HART on the 1769sc-OF4IH

HART pass through command request – command message packet structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HART Command</td>
<td>N bytes</td>
<td>The actual HART command PDU</td>
</tr>
<tr>
<td></td>
<td>N = Length of message – 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Contents are as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start or Delimiter (1 byte): 0x82</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Long form Address (5 bytes)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HART Command number (1 byte)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Request Data Count (1 byte)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data (“Request Data Count” bytes)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checksum (XOR of all bytes from delimiter on. Delimiter is included)</td>
<td></td>
</tr>
</tbody>
</table>

Table 6-12. HART Pass-Through Request Reply

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HART Channel</td>
<td>0x00 – 0x03 (1 byte)</td>
<td>Module input channel number for HART command</td>
</tr>
<tr>
<td>Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>(1 byte)</td>
<td>Command status</td>
</tr>
<tr>
<td></td>
<td>32 = Busy (Queue is already full).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>33 = DR_INITIATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35 = DR_DEAD (bad request)</td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td>(1 byte)</td>
<td>Set to 1</td>
</tr>
<tr>
<td>Handle</td>
<td>(1 byte)</td>
<td>The handle for command complete query</td>
</tr>
<tr>
<td></td>
<td>0 (bad when status is DR_DEAD)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1-255 (good)</td>
<td></td>
</tr>
</tbody>
</table>

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

INITIATE will be sent back under the following conditions:

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

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

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

**Table 6-13. HART Pass-Through Query Command**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HART Channel Number</td>
<td>0x00 – 0x03 (1 byte)</td>
<td>Module input channel number for HART command</td>
</tr>
<tr>
<td>Command Number</td>
<td>0x0C (1 byte)</td>
<td>The command number</td>
</tr>
<tr>
<td>Handle</td>
<td>(1 byte)</td>
<td>The handle from command request reply</td>
</tr>
<tr>
<td></td>
<td>1-255</td>
<td></td>
</tr>
</tbody>
</table>

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

**Table 6-14. HART Pass-Through Query Reply NOT SUCCESS**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconnected Message Header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HART Channel Number</td>
<td>0x00 – 0x07 (1 byte)</td>
<td>Module input channel number for HART command</td>
</tr>
<tr>
<td>Status</td>
<td>(1 byte)</td>
<td>Command status</td>
</tr>
<tr>
<td></td>
<td>34 = DR_RUNNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35 = DR_DEAD (bad request)</td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td>(2 bytes) (Command Number 0x0C)</td>
<td>Length of Handle + HART Response Data in</td>
</tr>
</tbody>
</table>
HART pass through command complete query - reply packet structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconnected Message Header</td>
<td>bytes (if Success)</td>
<td></td>
</tr>
<tr>
<td>Handle</td>
<td>(1 byte)</td>
<td>The handle from command complete query</td>
</tr>
</tbody>
</table>

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

Table 6-15. HART Pass-Through Query Reply SUCCESS

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HART Channel Number</td>
<td>0x00 – 0x07 (1 byte)</td>
<td>Module input channel number for HART command</td>
</tr>
<tr>
<td>Status</td>
<td>(1 byte)</td>
<td>00 = SUCCESS</td>
</tr>
<tr>
<td></td>
<td>00 = SUCCESS</td>
<td>Command status</td>
</tr>
<tr>
<td>Count</td>
<td>(1 byte)</td>
<td>Length of Handle + HART Response Data in bytes (if Success)</td>
</tr>
<tr>
<td></td>
<td>(Command Number 0x04)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2 bytes)</td>
<td>(Command Number 0x08, 0x0C)</td>
</tr>
<tr>
<td>Handle</td>
<td>1-255</td>
<td>The handle from command complete query</td>
</tr>
<tr>
<td>HART Command Response Data</td>
<td>Size is the entire HART device response size in bytes. The size does not include preambles bytes.</td>
<td>The HART device’s response to the command (if Success)</td>
</tr>
</tbody>
</table>

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.

**Figure 6-11a. Pass-Through Ladder**
Figure 6-11b.

The following rung loads the field device address and initializes the Pass-Through Request and Pass-Through Query commands.

```
InitializeDelay.DN

MOV
Source Channel 0
Dest 0f4i0PassThruReqTX[0] 0

MOV
Source 1
Dest 0f4i0PassThruReqTX[1] 1

MOV
Source -128
Dest 0f4i0PassThruReqTX[2] -128

MOV
Source 0f4i0Packet0[Channel,0].ManufacturerID
Dest 0f4i0PassThruReqTX[3] -66

MOV
Source 0f4i0Packet0[Channel,0].DeviceType
Dest 0f4i0PassThruReqTX[4] 2

COP
Source 0f4i0Packet0[Channel,0].DeviceSerialNumber[0]
Length 3
Dest 0f4i0PassThruReqTX[5] 3

MOV
Source Channel 0
Dest 0f4i0PassThruQryTX[0] 0

MOV
Source 12
Dest 0f4i0PassThruQryTX[1] 12
```
Figure 6-11c.

The following rung loads the HART command defined by the user. Note: You must populate the Command, ByteCount, and Data tag before setting the LoadCommand bit.

4

MOV

Move
Source
Command
44
Dest
Of4ih0PassThruReqTX[8]
44

MOV

Move
Source
ByteCount
1
Dest
Of4ih0PassThruReqTX[9]
1

COP

Copy File
Source
DATA[0]
Dest
Of4ih0PassThruReqTX[10]
Length
ByteCount

The following five rungs calculate the checksum for the HART message. The result is stored in the Checksum tag.

5

ADD

Add
Source A
8
Source B
ByteCount
1
Dest
XorControl.LEN
9

6

CPT

Compute
Dest
ChecksumPos
11
Expression
10+ByteCount

7

FAL

LoadCommand

File Arith/Logical
Control
Length
Mode
Dest
XorControl

Expression
0+Of4ih0PassThruReqTX[XorControl.POS+2] XOR(0+Checksum)
Figure 6-11d.

If the ExecuteMessage bit is set, this rung will send the HART command stored in the PassThruReqTX to the device.

The following rung checks for the initiate response code for the HART Pass-Through Command Request. If the initiate response code is received the current handle is copied over to the appropriate byte in the HART Pass-Through Command Query transmit tag.
When the "Initiate" response is received from the HART Command Request, this rung will send the HART Command Query to pull the data received by the module from the field device. The response data can be found in the PassThruQryRX tag.
Figure 6-11f.

This rung resets the logic used to perform the Pass-Through commands.

NOTE

The ladder in Figure 6-11 can be found in the project sample file located on our website at (www.spectrumcontrols.com)
Section 6.5
HART Protocol
Overview

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

6.5.1 Message Format

HART protocol specifies a message structure as follows:

**Figure 6-12. HART Message Structure**

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Start Character</th>
<th>Address</th>
<th>Command</th>
<th>Byte Count</th>
<th>Status</th>
<th>Data</th>
<th>Checksum</th>
</tr>
</thead>
</table>

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

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

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

**Preamble**

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

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

**Start Character**

The start character in a HART message has various values, indicating which frame format is being used, the source of the message, and whether a field device is in burst mode. The possible definitions are shown in the table below.
Table 6-16. Start Character Definition

<table>
<thead>
<tr>
<th></th>
<th>Short Frame</th>
<th>Long Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master to slave</td>
<td>02 (Hex)</td>
<td>82 (Hex)</td>
</tr>
<tr>
<td>Slave to master</td>
<td>06 (Hex)</td>
<td>86 (Hex)</td>
</tr>
<tr>
<td>Burst mode from slave</td>
<td>01 (Hex)</td>
<td>81 (Hex)</td>
</tr>
</tbody>
</table>

**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 format, we will omit the discussion of the short frame format. In either format, the single-bit address of the master is the most significant. Only two masters are allowed for example, a control system and a hand-held communicator. The most significant bit of the address field differentiates these two hosts. Primary masters such as the OF4IH use address 1, and secondary masters such as hand helds use address 0. Please see figure below.

**Figure 6-1. Long Frame Address**

![Diagram of Long Frame Address]

**NOTE**

The OF4IH does not support burst mode.

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

**Command**

The command byte contains an integer (0 to hex FF or decimal 257) that represents one of the HART commands. Code 254 is defined as an expansion code and is followed by another byte allowing more than 256 different commands to be defined if necessary. The received command code is echoed
back by the slave device in its reply.
There are three categories of commands: universal commands, which all HART
devices must implement; common practice commands, which should be used if
the associated function is provided; and device specific commands, which are for
functions unique to a specific slave device.

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

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

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

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

**6.5.2 Sending a HART Command to a Field Device via Pass-
through**
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 Of4ih0PassThruReqTX. This tag is
used in the ladder sample shown in Figure 6-.
<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Value in Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HART_PASS_THRU_QRY_RX[0]</td>
<td>00</td>
<td>HART channel</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[1]</td>
<td>00</td>
<td>Command Status</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[2]</td>
<td>15</td>
<td>Length of handle + HART response data (Byte 1)</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[3]</td>
<td>0</td>
<td>Length of handle + HART response data (Byte 2)</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[4]</td>
<td>02</td>
<td>Message handle</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[5]</td>
<td>86</td>
<td>Start character</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[6]</td>
<td>BE</td>
<td>Long address byte 0</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[7]</td>
<td>02</td>
<td>Long address byte 1</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[8]</td>
<td>0C</td>
<td>Long address byte 2</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[9]</td>
<td>77</td>
<td>Long address byte 3</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[10]</td>
<td>37</td>
<td>Long address byte 4</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[12]</td>
<td>0B</td>
<td>Byte count = 11 decimal</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[13]</td>
<td>00</td>
<td>Status Byte 0</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[14]</td>
<td>00</td>
<td>Status Byte 1</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[15]</td>
<td>20</td>
<td>Range units code = 32 decimal</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[16]</td>
<td>44</td>
<td>Upper Range value (This is a floating-point value = 600)</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[17]</td>
<td>16</td>
<td>Note: The bytes are in reverse order.</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[18]</td>
<td>00</td>
<td>Lower Range value (This is a floating-point value = -150)</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[19]</td>
<td>00</td>
<td>Note: The bytes are in reverse order.</td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[20]</td>
<td>C3</td>
<td></td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[21]</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[22]</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[23]</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>HART_PASS_THRU_QRY_RX[24]</td>
<td>F9</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

The HART message string, shown above, performs HART command 35 (write range values). Once the tags are populated with the HART message, the message can be sent using the ladder in Figure 6-. The reply for the HART command will be found in the Of4ih0PassThruQryRX tag. The response message should look like the table shown below.
### Table 6-18.

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

Programming Examples

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

Section 7.1

CompactLogix

7.1.1 Reset/Reconfig

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

Figure 7-1. Reset

![Diagram of reset/reconfiguration ladder sample]
Figure 7-2. Reconfig

This rung will reconfigure the OF41H without breaking the backplane connection. This rung should be used if the user wishes to reconfigure the module during system operation.

ReconfigEN

Type - Module Reconfigure
Message Control
ReconfigMsg

ReconfigMsg dn

ReconfigEN (U)

Message Configuration - ReconfigMsg

Configuration Communication Tag

Message Type: Module Reconfigure

Update module configuration without interrupting the connection.

Enable Enable Waiting Start Done Done Length 0

Error Code Error Path Error Text

OK Cancel Send Help

User’s Manual 0300217-04 Rev. A
7.1.3 Swap Byte Order
This ladder sample demonstrates how to reverse the order of the bytes for a floating-point tag and then convert it to 4 consecutive SINT tags, so that it can be used in a HART message.

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

Figure 7-3 Change Byte Order
The following rungs convert a real into its four byte HART equivalent.

7.1.4 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.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
</tr>
<tr>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>^</td>
<td>_</td>
</tr>
<tr>
<td>SP</td>
<td>!</td>
<td>*</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>´</td>
<td>(</td>
<td>)</td>
<td>*</td>
<td>+</td>
<td>,</td>
<td>.</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>:</td>
<td>;</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
<td>?</td>
</tr>
</tbody>
</table>

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.

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

**Figure 7-4a. Packed ASCII**

The following rungs set the starting bit address at 5 for the "unpacked ASCII" byte. This rung makes it possible to truncate bits 6 and 7 in

The following rungs determine which bit will be shifted from the "unpacked ASCII" byte to the double word containing the packed ASCII characters.

This rung causes the process to advance to the next byte in the unpacked ASCII string.
Figure 7-4b.

After all four bytes have been packed this rung stops the packing process, resets the ladder and copies the result into a 4 byte array (Result). Note: The first byte in the result is 0 and should be ignored.
Section 7.2
MicroLogix 1500

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

| NOTE | It is recommended that a 1764-LRP series C processor with firmware version 5 or higher be used. The LRP processor supports floating point files, which is required to read floating point data from the OF4IH. |
| Table 7-1. Routine Description |
| Routine | Description |
| MAIN | The main routine is the starting point for the ladder program. |
| PACKETS | The “packets” routine is used to demultiplex the HART data from the input file to individual integer files, so that the data can be viewed or used within the ladder program. This routine is called from the MAIN routine. |
| MSG_TO_MOD | This routine is used to send and receive messages to and from the module. Refer to Chapter 7 for more details regarding sending and receiving messages. This routine is called from the HART_MSG routine. |
| SRC_CHECK | Calculates the checksum for a message sent to the module one page at a time. This routine is called from the MSG_TO_MOD routine. |
| DEST_CHECKSUM | This routine calculates the checksum for a message received from the module one page at a time. This routine is called from the MSG_TO_MOD routine. |
| HART_MSG | This routine composes HART messages that will be sent to the module/field transmitter. This routine is called from the MAIN routine. |
| WORD_BYTE | Converts word data to its byte equivalent. This routine is called from the HART_MSG routine. |
| HART_CHECK | Calculates the checksum for the HART message being sent to the module/field device. This routine is called from the HART_MSG routine. |
| BYTE_WORD | Converts byte data to its word equivalent. This routine is called by the HART_MSG routine. |
7.2.1 MAIN Routine
The main routine is the starting point for the ladder program.

Figure 7-5. Main Routine

7.2.2 PACKETS Routine
The “packets” routine is used to demultiplex the HART data from the input file to individual integer files, so that the data can be viewed or used within the ladder program. This routine is called from the MAIN routine.
### Figure 7-6a. Packets Routine

The following rung determines the module channel number being scanned.

```
CURRENT_CHANNEL
NUMBER BEING
SCANNED
OF4H0_CH_NUMBER
AND
Bitwise AND
Source A 1:1.6 0210h<
Source B 3 1792<
Dest N7:0 0000h<
```

The following rung determines the current packet number for the HART data being received by the module.

```
CURRENT_PACKET
NUMBER BEING
SCANNED
OF4H0_PACKET_NUMBER
AND
Bitwise AND
Source A 1:1.6 0210h<
Source B 1792<
Dest N7:1 0001h<
```

```
CURRENT_PACKET
NUMBER BEING
SCANNED
OF4H0_PACKET_NUMBER
DIV
Divide
Source A N7:1 1<
Source B 256 256<
Dest N7:1 1<
```

Temporary holding register #1

```
TEMP1
ADD
Add
Source A N7:1 1<
Source B 10 10<
Dest N7:2 39<
```
The following rung copies the HART packet data from the input data file to integer files 10 through 14 for each of the four channels.

N10 = Packet0 = Device specific data (i.e., device serial number, hardware revision, etc.)
N11 = Packet1 = The four dynamic HART variables, unit codes, variable assignment codes, and output range
N12 = Packet2 = HART slot variables and associated unit codes
N13 = Packet3 = ASCII message stored on field device
N14 = Packet4 = Extended status if applicable

The following rung converts the floating point values, for the four dynamic variables stored in integer file N11, to floating point values stored in file F20.
Figure 7-6c.

The following rung converts floating point values for the output range of the connected field device, which are stored in integer file N11, to floating point values stored in file F20.

0008
Temporary holding register #1
TEMP1
ADD
Source A
N7:2
Source B
N7:3
Dest
N7:2

0009
Temporary holding register #3
TEMP3
ADD
Source A
N7:4
Source B
4
Dest
N7:4

0010
#F20 [TEMP3]
CPW
Source #N11: [N7:2]
Dest #F20: [N7:4]
Length 4

0011
END
7.2.3 MSG_TO_MOD Routine

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

**Figure 7-7a. Message to Module**

The following runs resets the paging logic within the HART module to make ready for the next message.

```
0000
  RESET MESSAGE TO
  MODULE LADDER
  OF4H0_RESET
  B3:0
  0

0001
  EQU
  Source A N7.5 0<
  Source B 0 0<
```

```
0010
  CLEAR
  Dest N7.5 0<

DATA READ FROM
MODULE
#OF4H0_DEST_DATA
FILL
Fill File
Source 0
Dest #N15:0
Length 128

OF4H0_MSG_ER
B3:0
0<

1
OF4H0_MSG_DN
B3:0
0<

2
RESET MESSAGE TO
MODULE LADDER
OF4H0_RESET
B3:0
0

MSG_MASTER_CONTROL
CLEAR
CLR
Clear
Dest O:1.6 257<
```
Figure 7-7b.
Figure 7-7c.

The following rungs copy the current page of data over to the module via the output data file.

**MESSAGE TO MODULE**

**CURRENT STEP**

**OF4I66_STEP**

**EQU**

**NUMBER OF SOURCE**

**BYTES REMAINING**

**OF4I66_SRC_REMAINING**

**SUB**

**003**

Equal

Source A  N7:5

Source B  1<

**004**

Equal

Source A  N7:5

Source B  1<

**Subtract**

Source A  O:1.7

Source B  N7:7

Dest  N7:10

3<

**XOR CHECKSUM FOR**

**SOURCE DATA**

**POSITION**

**OF4I66_SRC_CHECK_POS**

**MOV**

Move

Source  N7:6

Dest  N7:11

0<

**Math**

**Register**

**DIV**

Divide

Source A  N7:6

Source B  2

Dest  S:15

2<

1<

**Temporary holding**

**register #1**

**TEMP1**

**MOV**

Move

Source  S:14

Dest  N7:2

8<

39<
Figure 7.7d.

If the bytes remaining are less than 38, this ring copies the data for the current page being sent to the message output buffer. This ring also calculates the data index of the message.
Figure 7-7e.

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

---

**MESSAGE TO MODULE CURRENT STEP**

**OF4H0 STEP**

**NUMBER OF SOURCE BYTES REMAINING**

**OF4H_SRC_REMAINING**

**NUMBER OF SOURCE BYTES REMAINING**

**OF4H_SRC_REMAINING**

---

0006

<table>
<thead>
<tr>
<th>EQU</th>
<th>NEQ</th>
<th>GEQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal</td>
<td>Source A</td>
<td>N7:5</td>
</tr>
<tr>
<td>Source B</td>
<td>N7:10</td>
<td></td>
</tr>
<tr>
<td>0&lt;</td>
<td>Source B</td>
<td>3&lt;</td>
</tr>
<tr>
<td>1&lt;</td>
<td>Source B</td>
<td>0</td>
</tr>
<tr>
<td>1&lt;</td>
<td>Source B</td>
<td>0&lt;</td>
</tr>
<tr>
<td>Grtr Thn or Eqv (A&gt;B)</td>
<td>Source A</td>
<td>N7:10</td>
</tr>
<tr>
<td>Source A</td>
<td>3&lt;</td>
<td></td>
</tr>
<tr>
<td>Source B</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>Source B</td>
<td>38&lt;</td>
<td></td>
</tr>
</tbody>
</table>

---

Message Request Buffer

Message #OF4H0_REQ_BUFFER

Copy File

Source #16[N7:2]

Dest #0:18

Length 19

---

**SOURCE BYTES SENT**

**OF4H0 SOURCE SENT**

**ADD**

**Source A**

**Source B**

**Dest**

N7:7

3<

38

38<

N7:7

3<

---

**SOURCE DATA**

**TEMPORARY INDEX**

**OF4H0_SRC_TMP_INDEX**

**MOV**

**Move**

**Source**

38

38<

**Dest**

N7:12

3<

---

**SOURCE DATA INDEX**

**OF4H0 SOURCE_INDEX**

**MOV**

**Move**

**Source**

N7:7

3<

**Dest**

N7:6

3<

---

**XOR CHECKSUM FOR SOURCE DATA LENGTH**

**OF4H0_SRC_CHECK_LEN**

**MOV**

**Move**

**Source**

N7:7

3<

**Dest**

N7:13

3<
Figure 7-7f.

The following rung executes a routine which calculates the checksum for the current page of data.

```
MESSAGE TO MODULE
CURRENT STEP
OF4IHO_STEP
EQU

Equal
Source A  N7:5
0<
Source B  1
1<
```

```
JSR
Jump To Subroutine
SBR File Number  U:5
```

The following rung copies the checksum value for the current page to the last word at the end of the page (i.e. <= Word 20).

```
MESSAGE TO MODULE
CURRENT STEP
OF4IHO_STEP
EQU

Equal
Source A  N7:5
0<
Source B  1
1<
```

```
DIV
Divide
Source A  N7:12
3<
Source B  2
2<
Dest  S.13
1<
```

```
ADD
Add
Source A  S.14
8<
Source B  8
8<
Dest  N7:2
39<
```

```
TEMP1
Temporary holding register #1
```

```
```

Figure 7-7g.

Check for message corruption.

15
EQU
Source A 04h00Msg Step 0
Source B 4
AND
Bitwise AND
Source A 04h00Input_MsgSlaveControl 16#0101
Source B 255
Dest Temp1 256

16
EQU
Source A 04h00Msg Step 0
Source B 4
AND
Bitwise AND
Source A 04h00Input_MsgSlaveControl 16#0101
Source B 255
Dest Temp1 256

The following rung sets the position of the pointer used to calculate the checksum for the current page of data being received.

17
EQU
Source A 04h00Msg Step 0
Source B 5
SUB
Subtract
Source A 04h00Input_MsgResponseSize 18
Source B 04h00Msg.DestBytesSent 18
Dest 04h00Msg.DestBytesRemaining 18

18
EQU
Source A 04h00Msg Step 0
Source B 5
MOV
Move
Source 04h00Msg.DestDataIndex 18
Dest Temp2 0

Wait for next response page.

O4h00Msg.Ed
End Now
JMP

The following rung calculates the bytes remaining for the current message being received.

MOV
Source 5
Dest 04h00Msg Step 0
Figure 7-7h.

The following rungs determine if the module accepted the data contained in the page last sent.

**Message to Module**

**Current Step**

**OF4H0_STEP**

**NEQ**

Not Equal

Source A N7:5

Source B 1

1<

**Source Checksum**

**OF4H0_SOURCE_CHECK**

**CLR**

Clear

Dest N7:14

0<

**XOR Checksum for**

**Source Data**

**(POSITION)**

**OF4H0_SRC_CHECK_POS**

**CLR**

Clear

Dest N7:11

0<

**Message to Module**

**Current Step**

**OF4H0_STEP**

**EQU**

Equal

Source A N7:5

0<

Source B 2

2<

**Temporary Holding**

**Register #1**

**TEMP1**

**AND**

Bitwise AND

Source A 1.1.26

0101<

Source B 255

255<

Dest N7:2

0027<

**Message to Module**

**Current Step**

**OF4H0_STEP**

**EQU**

Equal

Source A N7:2

39<

Source B 255

255<

**Temporary Holding**

**Register #1**

**TEMP1**

**DIV**

Divide

Source A 1.1.26

257<

Source B 256

256<

Dest N7:2

39<

**Message to Module**

**Current Step**

**OF4H0_STEP**

**EQU**

Equal

Source A N7:2

39<

Source B N7:8

1<

**MOV**

Move

Source 3

3<

Dest N7:5

0<
Figure 7-71.

The following rung checks to see if the current page is the last page in the message being sent to the module. If not, the logic jumps to rung 3 and the process repeats.

If the last page in the message was sent, this rung enables the module to start sending data back to the PLC starting with the first page.
The following runs copy the first page of data from the module to the PLC.

**Figure 7-7j.**

```
0015
  EQU
  Equal
  Source A N7:5 0<
  Source B 4 4<
```

```
0016
  EQU
  Equal
  Source A N7:5 0<
  Source B 4 4<
```

```
Temporary holding
register #1
TEMP1
```

```
OF4I0_MSGER
B3:0
39<
1
ENDNOW
Q4:2
DMP
```

```
MESSAGE TO MODULE
CURRENT STEP
OF4I0_STEP
```

```
Destination data
PAGE INDEX
OF4I0_DEST_PG_INDEX
EQU
Equal
Source A N7:17 1<
Source B N7:2 39<
```

```
MESSAGE TO MODULE
CURRENT STEP
OF4I0_STEP
```

```
Move
Source 5 5<
Dest N7:5 0<
```

```
AND
Bitwise AND
Source A 1:1.26 0101b<
Source B 255 255<
Dest N7:2 0027h<
```

```
Temporary holding
register #1
TEMP1
```

```
OF4I0_MSGER
B3:0
39<
1
ENDNOW
Q4:2
DMP
```

```
MESSAGE TO MODULE
CURRENT STEP
OF4I0_STEP
```

```
Move
Source 5 5<
Dest N7:5 0<
```
Figure 7-7k.
If the bytes remaining are less than 38, this rung copies the data from the message response buffer to a temporary array were the data can be viewed. This rung also calculates the data index.
Figure 7-7m.

If the bytes remaining are greater than or equal to 38, this rung copies the data from the message response buffer to a temporary array were the data can be viewed. This rung also calculates the data index.
Figure 7-7n.

The following rung calculates the checksum for the current page sent from the module to the PLC.

```
MESSAGE TO MODULE
CURRENT STEP
OF4H0_STEP
EQU
Equal
Source A N7:5
Source B 5
0<
5<

ISR
Jump To Subroutine
SBR File Number U:6
```

The following rungs verify if the checksum calculated by this ladder program for the current page are equal to the checksum provided by the module for the same page.

```
MESSAGE TO MODULE
CURRENT STEP
OF4H0 STEP
EQU
Equal
Source A N7:5
Source B 5
0<
5<

DIV
Divide
Source A N7:20
Source B 2
17<
2<

Dest S:13
1<

ADD
Add
Source A S:14
Source B 28
8<
28<

Dest N7:2
39<
```
Figure 7-7o.

MESSAGE TO MODULE
CURRENT STEP
OF4H0_STEP

EQU
Equal
Source A  N7:5
0<
Source B  5
5<

Temporar[y holding
register #1
TEMP1

ADD
Add
Source A  N7:2
39<
Source B  1
1<
Dest       N7:2
39<

Temporar[y holding
register #2
TEMP2

MOV
Move
Source  1:1[N7:2]
0<
Dest       N7:3
0<

Temporar[y holding
register #2
TEMP2

Math
Register

EQU
Equal
Source A  8:13
1<
Source B  0
0<

OF4H0_MSG_ER

B3:0
σ

OF4H0_DEST_CHECK

NEQ
Not Equal
Source A  N7:22
0<
Source B  N7:3
0<

DESTINATION DATA
XOR CHECKSUM
OF4H0_DEST_CHECK

ENDNOW
Q4:2
IMP

Multipl[y
Multiply
Source A  N7:17
1<
Source B  256
256<
Dest       N7:2
39<

User's Manual 0300217-04 Rev. A
Figure 7-7q.
The following rung determines if the current page is the last page being sent from the module to the PLC. If not, the ladder jumps to rung 15 and the process repeats.

```
0025
EQU
Equal
Source A  N7:5
Source B  6
N7:5
6

0026
EQU
Equal
Source A  N7:5
Source B  6
N7:5
6
```

If this is the last page of the message being sent from the module to the PLC, then flag the done bit and return to the calling routine.

```
0026
GEQ
Greater Than or Eq (A>=B)
Source A  N7:16
Source B  1.127
N7:16
1.127

0027
EQU
Q4:2
```

```
0028
LABEL
Return
```

7.2.4 SRC_CHECK Routine

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

**Figure 7-8a. Source Checksum**

The following rungs calculate the checksum for the current page of data being sent to the module from the PLC.

```
DEC
   DIV
   Source A  N7:13
   Source B  2<
   Dest      S:13
   1<
   Temporary holding
   register #1
   TEMP1
   MOV
   Source   S:14
   Temp     S<
   Dest     N7:2
```

```
0000

Math Register

NEQ
   Not Equal
   Source A  S:13
   Source B  0<
   0<

N16[TEMP1]
   AND
   Bitwise AND
   Source A  N16[N7:2]
   Source B  255
   Dest      N16[N7:2]
   0000b<

0001

XOR CHECKSUM FOR
SOURCE DATA
(LENGTH)
OF4H0_SRC_CHECK_LEN

EQU
   Equal
   Source A  N7:13
   Source B  0<
   0<

RET
   Return

0002

LOOP Q5:1
   LBL

XOR CHECKSUM FOR
SOURCE DATA
(POSITION)
OF4H0_SRC_CHECK_POS

GRT
   Greater Than (A>B)
   Source A  N7:11
   Source B  254
   254<

0003

XOR CHECKSUM FOR
SOURCE DATA
(POSITION)
OF4H0_SRC_CHECK_POS

MOV
   Move
   Source   0<
   Dest     N7:11
   0<
```
Figure 7-8b

Math Register
- DIV
  - Divide
  - Source A: N7:11
  - Source B: 2
  - Dest: S:13

Temporary holding register #1
- MOV
  - Move
  - Source: S:14
  - Dest: N7:2

SOURCE CHECKSUM FOR SOURCE DATA (POSITION)
- XOR
  - Bitwise Exclusive OR
  - Source A: N16[N7:2]
  - Source B: N7:14
  - Dest: N7:14

XOR CHECKSUM FOR SOURCE DATA
- ADD
  - Add
  - Source A: N7:11
  - Source B: 2
  - Dest: N7:11

Grtr than or Eq (A>=B)
- Source A: N7:11
  - Source B: N7:13

Loop
- Q5-1
- JMP

Return
7.2.5 DEST_CHECKSUM Routine

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

Figure 7-9a Destination Checksum

The following routine calculates the checksum for the current page of data being sent from the module to the PLC.

```
DESTINATION DATA
XOR CHECKSUM
(LENGTH)
OF4H0 DEST_XOR_LEN

0000
LOOP2 Q6:1 LBL

Equal
Source A N7:21
Source B 0

0001
DESTINATION DATA
XOR CHECKSUM
(POSITION)
OF4H0 DEST_XOR_POS

Greater Than (A>B)
Source A N7:19
Source B 254

0002
DESTINATION DATA
XOR CHECKSUM
(MOVE)
OF4H0 DEST_XOR_POS

Math
Register

DIV
Divide
Source A N7:19
Source B 2
Dest S:13

Next
Temporary holding
Register #1
TEMP1

MOV
Move
Source S:14
Dest N7:2
```
### 7.2.6 HART_MSG Routine

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

**Figure 7-10a HART Message**

- **SELECT TARGET CHANNEL FOR MESSAGE**
  - **LE5**
    - Less Than (A>B)
    - Source A: N7:23
    - Source B: 0
  - **GE5**
    - Greater Than (A>B)
    - Source A: N7:23
    - Source B: 3

- **LOAD_COMMAND**
  - B3:0
  - 7

- **EXECUTE_MESSAGE**
  - B3:0
  - 3

- **INITIALIZE_DELAY**
  - Timer On Delay: T4:0
  - Time Base: 1.0
  - Preset: 5
  - Accum: 5

- **MUL**
  - Multiply
  - Source A: N7:23
  - Source B: 20
  - Dest: N7:2

- **COP**
  - Copy File
  - Source: #N10-[N7:2]
  - Dest: #N16:0

- **WORD_BYTE_LEN**
  - Move
  - Source: 20
  - Dest: N7:30

- **MESSAGE_PENDING**
  - B3:0
  - 7

- **JSR**
  - Jump To Subroutine
  - SHR File Number: U:8
Figure 7-10b

MESSAGE_PENDING
BX0
7

COP
Copy File
Source
#N15.0
Dest
#57.40
Length
40
Figure 7-10c

```
<table>
<thead>
<tr>
<th>INITIALIZE_DELAY/DN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4:0</td>
</tr>
<tr>
<td>DN</td>
</tr>
</tbody>
</table>

MOV

- Source: N7:23
- Dest: N30:0

MOV

- Source: 1
- Dest: N30:1

MOV

- Source: 130
- Dest: N30:2

MOV

- Source: N7:42
- Dest: N30:3

MOV

- Source: 128

MOV

- Source: N7:43
- Dest: N30:4

COP

- Source: #N7:52
- Dest: #N30:5
- Length: 3
```
Figure 7-10d

The following rung loads the HART command defined by the user. Note: You must populate the command, byte count and data tag before setting the bit.

```
LOAD_COMMAND
B3:0
```

```
MOV
Source  N7:23
Dest    N32:0
```

```
MOV
Source  12
Dest    N32:1
```

```
MOV
Source  N7:25
Dest    N30:8
```

```
MOV
Source  N7:26
Dest    N30:9
```

```
COP
Copy File
Source  #N9:0
Dest    #N30:10
Length  50
```

```
LENGTH OF HART MESSAGE
HART_CHECK_LEN
ADD
Add
Source A  8
8<
Source B  N7:26
1<
Dest     N7:27
9<
```
The following ring executes a routine which calculates the checksum for the current HART message being sent to the module.

LOAD_COMMAND

ADD

HART_CHECKSUM
POSITION
HART_CHECK_POS

Add
Source A N7:26
Source B 10
Dest N7:28

JSR
Jump To Subroutine
SBR File Number U:9

N30[HART_CHECK_POS]

MOV
Move
Source N7:29
Dest N30[N7:28]

EXECUTE_MESSAGE

LOAD_COMMAND

CHECKSUM FOR CURRENT
HART MESSAGE
HART_CHECKSUM

CLR
Clear
Dest N7:29

Figure 7-10f
Figure 7-10g

If the execute message tag is set, this rung will send the HART command stored in the Req_TX to the device. Note: This command will continue to reset until the Request Initiate, Request Dead, or query dead is received.

EXECUTE MESSAGE REQUEST INITIATE

MESSAGE_PENDING

ADD

Source A N7.28
Source B 1
Dest N7.9

COPY

Source #N15:0
Dest #N16:0
Length 128

RESET MESSAGE TO MODULE LADDER
OF4H0 RESET
B3:0
(1)
0

ISR
Jump To Subroutine
SBR File Number U:4
The following rung checks for the initiate response code for the HART Pass-Through Command Request. If the initiate response code is received, the current handle is copied over to the appropriate byte in the HART Pass-Through Command Query transmit tag.

**Figure 7-10h**

```
0017
  EQU
  Equal Source A N31:1 0<
  Source B 33 33<

0018
  EQU
  Equal Source A N31:1 0<
  Source B 32 32<

0019
  EQU
  Equal Source A N31:1 0<
  Source B 35 35<
```
Figure 7-10i

Convert integer data from byte to word format. Note: Upper byte is padded with a zero.

When the "Initiate" response is received from the HART Command Request, this rung will send the HART Command Query to pull the data received by the module from the field device. The response data can be found in QRY_RX (i.e. File N33). Note: This rung will continue to execute until the Query Success, Query Dead or Request Dead is received.

User's Manual 0300217-04 Rev. A
Figure 7-10j

The following rung copies the response from the HART transmitter to file N33 QRY_RX.

EXECUTE_MESSAGE OF4H0_MSG_DN

```
0022

<table>
<thead>
<tr>
<th>B3:0</th>
<th>B3:3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
```

```
COP
Copy File
Source #N15:0
Dest #N16:0
Length 128

WORD_BYTE_LEN

MOV
Move
Source I:1.27
Dest N7:30

JSR
Jump To Subroutine
SBR File Number U:8

FILL
Fill File
Source 0
Dest #N33:0
Length 128

FILL
Fill File
Source 0
Dest #N33:128
Length 128

COP
Copy File
Source #N15:0
Dest #N33:0
Length 128

COP
Copy File
Source #N15:128
Dest #N33:128
Length 128

OF4H0_MSG_DN

```

```
0025

<table>
<thead>
<tr>
<th>B3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
```

```
EQU
Equal
Source A N33:1
Source B 0

QUERY_SUCCESS

```

User's Manual 0300217-04 Rev. A
Figure 7-10k

The following rung resets the process and makes ready for the next HART message.

```
0027
RESET_HART_MESSAGE
B3:0
13

REQUEST_HART_MESSAGE
B3:0
13

EXECUTE_MESSAGE
EXECUTE_MESSAGE
B3:0
1
4

EXECUTE_MESSAGE
B3:0
1
4

MOV
Move
Source
0
Dest
N3:1
0

MOV
Move
Source
1
Dest
N3:1
1
```
7.2.7 WORD_BYTE Routine

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

**Figure 7-11a Word to Byte**

The following routine converts word data contained in an integer file (i.e. N15) into its byte equivalent. The result is placed in a second integer file (i.e. N16)
Figure 7-11b

```
Figure 7-11b

N15[TEMP1]
AND
Bitwise AND
Source A N16[N7:24]
0000h<
Source B 255
255<
Dest N15[N7:2]
0000h<

Temporary holding register #1
TEMP1
ADD
Add
Source A N7:2
39<
Source B 1
1<
Dest N7:2
39<

Temporary holding register #2
TEMP2
AND
Bitwise AND
Source A N16[N7:24]
0000h<
Source B -256
-256<
Dest N7:3
0000h<

Temporary holding register #2
TEMP2
DIV
Divide
Source A N7:3
0<
Source B 256
256<
Dest N7:3
0<

N15[TEMP1]
AND
Bitwise AND
Source A N7:3
0000h<
Source B 255
255<
Dest N15[N7:2]
0000h<
```
Figure 7-11c

```
0003
Temporary holding register #4
WORD_BYTE_INDEX

0004
GEQ
Greater Than or Eq (A>=B)
Source A  N7:24
          20<
Source B  N7:30
          20<

0005
RET
Return

0006
LOOP3
Q8:10
(END)
```

Temporary holding
register #4
WORD_BYTE_INDEX

ADD
Add
Source A  N7:24
          20<
Source B  1
          1<
Dest      N7:24
          20<
HART_CHECK Routine
Calculates the checksum for the HART message being sent to the module/field device. This routine is called from the HART_MSG routine.

Figure 7-12a HART Checksum

The following routine calculates the checksum for the current HART message being sent to the module/field device.

```
LENGTH OF HART MESSAGE
LENGTH OF HART MESSAGE
GRT

Greater Than (A>B)
Source A N7:27
Source B 254

EQU

Length of HART_CHECK_LEN

MOV

Move
Source 0
Dest N7:27

ADD
Add
Source A N7:2
Source B 2
Dest N7:3
```

```
0000

LOOP4 Q9:11

LABEL

LENGTH OF HART MESSAGE
LENGTH OF HART MESSAGE
GRT

Greater Than (A>B)
Source A N7:27
Source B 254

EQU

Length of HART_CHECK_LEN

MOV

Move
Source 0
Dest N7:27

ADD
Add
Source A N7:2
Source B 2
Dest N7:3
```
Figure 7-12b

```
0004
0005
0006
0007

 CHECKSUM FOR CURRENT
 HART MESSAGE
 HART_CHECKSUM
XOR

 Bitwise Exclusive OR
 Source A N30:N7:3
     0000h<
 Source B N7:29
     0000h<
 Dest   N7:29
     0000h<

 Temporary holding
 register #1
 TEMP1
 ADD
 Add
 Source A N7:2
 39<
 Source B 1
 1<
 Dest   N7:2
 39<

 TEMP1
GEQ
Greater Than or Equal (A>B)
 Source A N7:2
 39<
 Source B N7:27
 9<

 HART_CHECK_DN
 B3:0
 Loop 4
 09:11
 JMP

 END
```

7.2.8 BYTE_WORD Routine

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

Figure 7-13a Byte to Word

The following routine converts byte data contained in an integer file (i.e. N15) into its word equivalent. The result is placed in a second integer file (i.e. N16).
Figure 7-13b

| 0005 | Temporary holding register #2 TEMP2 |
| 0006 | MUL |
| | Multiply Source A N16[N7:31] 0< |
| | Source B 256 |
| | 256< |
| | Dest N7:3 0< |
| 0007 | N15[TEMP1] |
| | MVN |
| | Masked Move Source N7:3 0< |
| | Mask 0FF00h |
| | -256< |
| | Dest N15[N7:2] 0< |

BYTE WORD POS

| 0008 | ADD |
| | Add Source A N7:31 3< |
| | Source B 1 1< |
| | Dest N7:31 3< |

LOOP5 Q10:12

| 0009 | END |
Chapter 8
Diagnostics and Troubleshooting

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

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

Section 8.1
Safety Considerations

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

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

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

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

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

8.1.3 Program Alteration
There are several possible causes of alteration to the user program, including extreme environmental conditions, Electromagnetic Interference (EMI), improper grounding, improper wiring connections, and unauthorized tampering. If you suspect a program has been altered, check it against a previously saved master program.
8.1.4 Safety Circuits
Circuits installed on the machine for safety reasons, like over-travel limit switches, stop push buttons, and interlocks, should always be hard-wired to the master control relay. These devices must be wired in series so that when any one device opens, the master control relay is de-energized, thereby removing power to the machine. Never alter these circuits to defeat their function. Serious injury or machine damage could result.

Section 8.2 Module Operation vs. Channel Operation

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

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

Section 8.3 Power-up Diagnostics

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

<table>
<thead>
<tr>
<th>If module status LED is:</th>
<th>Indicated condition</th>
<th>Corrective action:</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Proper Operation</td>
<td>No action required</td>
</tr>
<tr>
<td>Off</td>
<td>Module Fault</td>
<td>Cycle power. If condition persists, replace the module. Call your local distributor or Spectrum Controls for assistance.</td>
</tr>
</tbody>
</table>

Table 8-1 LED Status
Section 8.4
Channel Diagnostics

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

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

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

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

Section 8.5
Non-critical vs. Critical Module Errors

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

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

Section 8.6
Module Error Definition Table

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

<table>
<thead>
<tr>
<th>&quot;Don't Care&quot; Bits</th>
<th>Module Error</th>
<th>Extended Error Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12</td>
<td>11 10 09 08 07 06 05 04</td>
<td>03 02 01 00</td>
</tr>
<tr>
<td>00 00 00 00</td>
<td>00 00 00 00 00 00 00 00</td>
<td>00 00 00</td>
</tr>
</tbody>
</table>

Table 8-2 Module Error Table

Hex Digit 4 | Hex Digit 3 | Hex Digit 2 | Hex Digit 1
8.6.2 Module Error Field

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

Table 8-3 Module Error Types

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

8.6.3 Extended Error Information Field

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

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

Hardware Errors

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

Configuration Errors

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

Table 8-4 Extended Fault Codes lists the possible module-specific configuration error codes defined for the module.
## Section 8.7 Error Codes

The table below explains the extended error code.

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

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

## Section 8.8

### Module Inhibit Function

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

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

## Section 8.9

### Getting Technical Assistance

Note that your module contains electronic components which are susceptible to damage from electrostatic discharge (ESD). An electrostatic charge can accumulate on the surface of ordinary plastic wrapping or cushioning material.

In the unlikely event that the module should need to be returned to Spectrum Controls, please ensure that the unit is enclosed in approved ESD packaging (such as static-shielding / metallized bag or black conductive...
container). Spectrum Controls reserves the right to void the warranty on any unit that is improperly packaged for shipment.

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

**USA** - 425-746-9481

### Section 8.10 Declaration of Conformity

Available upon request
# Appendix A
## Module Specifications

### Electrical Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Channels</td>
<td>4 single isolated outputs with HART</td>
</tr>
<tr>
<td>Output Ranges</td>
<td>0 to 20.0 mA and 4 to 20 mA</td>
</tr>
<tr>
<td>Output Load</td>
<td>750 ohms maximum</td>
</tr>
<tr>
<td>Resolution</td>
<td>16 bits over full design range (0-21.24 mA)</td>
</tr>
<tr>
<td>Full Scale Settling Time</td>
<td>&lt;45 ms to 95% of full scale</td>
</tr>
<tr>
<td>without HART enabled</td>
<td></td>
</tr>
<tr>
<td>Full Scale Settling Time</td>
<td>&lt;70 ms to 95% of full scale</td>
</tr>
<tr>
<td>with HART enabled</td>
<td></td>
</tr>
<tr>
<td>Analog Output Update Rate</td>
<td>10 ms update of all 8 channels</td>
</tr>
<tr>
<td>Differential Nonlinearity</td>
<td>±0.05% full scale based on 4-20 mA range.</td>
</tr>
<tr>
<td>Output Ripple</td>
<td>±0.05%, or better full scale 0-50 kHz</td>
</tr>
<tr>
<td>Repeatability</td>
<td>±0.05% full scale</td>
</tr>
<tr>
<td>Max Inductive Load</td>
<td>0.1 mH</td>
</tr>
<tr>
<td>Output Impedance</td>
<td>&gt; 1 Mohm</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Temperature</td>
</tr>
<tr>
<td></td>
<td>Accuracy (% Full Scale)</td>
</tr>
<tr>
<td>Temperature</td>
<td>25 °C</td>
</tr>
<tr>
<td></td>
<td>0.35%</td>
</tr>
<tr>
<td>0 °C to 60 °C</td>
<td>0.55%</td>
</tr>
<tr>
<td>Calibration</td>
<td>Modules will be factory-calibrated.</td>
</tr>
<tr>
<td>Isolation Voltage</td>
<td>710 VDC terminal block to back plane continuous</td>
</tr>
<tr>
<td></td>
<td>710 VDC channel to channel continuous</td>
</tr>
<tr>
<td></td>
<td>710 VDC terminal block to frame ground continuous</td>
</tr>
<tr>
<td>Maximum Backplane Current draw</td>
<td>180 mA at 5 VDC, 200 mA at 24 VDC</td>
</tr>
<tr>
<td>Thermal Dissipation</td>
<td>4.2 W at 21.0 mA current all channels with 250-ohm minimum load.</td>
</tr>
<tr>
<td>Maximum Slot Distance</td>
<td>8 for backplane +24 V</td>
</tr>
<tr>
<td>Overvoltage Protection</td>
<td>+24V continuous on any current output</td>
</tr>
<tr>
<td>Power on/off channel glitch</td>
<td>Under ±1 V for 2 ms. Target: ±0.5 V spike for &lt;5 ms</td>
</tr>
</tbody>
</table>
### Appendix A: Module Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open and Short Circuit Protection</td>
<td>Yes, continuous. (EN61131-2 requirement)</td>
</tr>
</tbody>
</table>

### Environmental Specifications

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

<table>
<thead>
<tr>
<th>Regulatory Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>cULus listed to:</td>
</tr>
<tr>
<td>UL 508</td>
</tr>
<tr>
<td>ANSI/ISA 12.12.01 Class 1, Div2 Group A, B, C, D (CSA Equiv.)</td>
</tr>
<tr>
<td>CE compliance to EN 61010-1 and EN 61131-2, EN61000-6-2, EN61000-6-4</td>
</tr>
<tr>
<td>EN60079-0, EN60079-15 ATEX</td>
</tr>
</tbody>
</table>

![ATEX Certification](image)
ATEX Special Conditions for Safe Use

- Provision shall be made to prevent the rated voltage being exceeded by the transient disturbances of more than 140% of the peak rated voltage.
- The equipment shall be installed in an enclosure that provides a degree of protection not less than IP 54 in accordance with EN 60079-0 and used in an environment of not more than pollution degree 2. The enclosure shall be accessible only with the use of a tool.
- The enclosure must have a door or cover accessible only by the use of a tool.
Appendix B
HART Universal and Common Practice Commands

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

<table>
<thead>
<tr>
<th>COMMAND NUMBER AND FUNCTION</th>
<th>DATA IN COMMAND</th>
<th>(TYPE)</th>
<th>DATA IN REPLY</th>
<th>(TYPE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>38 Read transmitter variables</td>
<td>Byte 0</td>
<td>transm. var. code for slot 0</td>
<td>Byte 0</td>
<td>transmitter var. code for slot 0</td>
</tr>
<tr>
<td></td>
<td>Byte 1</td>
<td>transm. var. code for slot 1</td>
<td>Byte 1</td>
<td>units code for slot 0</td>
</tr>
<tr>
<td></td>
<td>Byte 2</td>
<td>transm. var. code for slot 2</td>
<td>Byte 2-5</td>
<td>variable for slot 0</td>
</tr>
<tr>
<td></td>
<td>Byte 3</td>
<td>transm. var. code for slot 3</td>
<td>Byte 6</td>
<td>transmitter var. code for slot 1</td>
</tr>
<tr>
<td></td>
<td>Byte 7</td>
<td>units code for slot 1 (truncated after last requested code)</td>
<td>Byte 8-11</td>
<td>variable for slot 1</td>
</tr>
<tr>
<td></td>
<td>Byte 12</td>
<td>transmitter var. code for slot 2</td>
<td></td>
<td>(F)</td>
</tr>
<tr>
<td></td>
<td>Byte 13</td>
<td>units code for slot 2 variable for slot 2</td>
<td></td>
<td>(F)</td>
</tr>
<tr>
<td></td>
<td>Byte 14-17</td>
<td>transmitter var. code for slot 3</td>
<td></td>
<td>(F)</td>
</tr>
<tr>
<td></td>
<td>Byte 18</td>
<td>units code for slot 3 variable for slot 3</td>
<td></td>
<td>(F)</td>
</tr>
<tr>
<td></td>
<td>Byte 19</td>
<td>(truncated after last requested variable)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 20-23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34 Write damping value</td>
<td>Byte 0-3</td>
<td>damping value (sec)</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>35 Write range values</td>
<td>Byte 0</td>
<td>range units code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1-4</td>
<td>upper range value</td>
<td>(F)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5-8</td>
<td>lower range value</td>
<td>(F)</td>
<td></td>
</tr>
<tr>
<td>36 Set upper range value (+ push SPAN button)</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>37 Set lower range value (+ push ZERO button)</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>38 Reset “configuration changed” flag</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>39 EEPROM control</td>
<td>Byte 0</td>
<td>EEPROM control code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0 = burn EEPROM, 1 = copy EEPROM to RAM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40 Enter/exit fixed current mode</td>
<td>Byte 0-3</td>
<td>current (mA)</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0 = exit the mode)</td>
<td>(F)</td>
<td></td>
</tr>
<tr>
<td>41 Perform transmitter self test</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>42 Perform master reset</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>43 Set PV zero</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>44 Write PV units</td>
<td>Byte 0</td>
<td>PV units code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>45 Trim DAC zero</td>
<td>Byte 0-3</td>
<td>measured current (mA)</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>46 Trim DAC gain</td>
<td>Byte 0-3</td>
<td>measured current (mA)</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>47 Write transfer function</td>
<td>Byte 0</td>
<td>transfer function code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>48 Read additional transmitter status</td>
<td>none</td>
<td>Byte 0-24</td>
<td>additional status</td>
<td></td>
</tr>
<tr>
<td>49 Write PV sensor serial number</td>
<td>Byte 0-2</td>
<td>sensor serial number</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>50 Read dynamic variable assignments</td>
<td>none</td>
<td>Byte 0</td>
<td>PV transmitter variable code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Byte 1</td>
<td>SV transmitter variable code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Byte 2</td>
<td>TV transmitter variable code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Byte 3</td>
<td>PV transmitter variable code</td>
<td></td>
</tr>
<tr>
<td>COMMAND NUMBER AND FUNCTION</td>
<td>DATA IN COMMAND</td>
<td>DATA IN COMMAND (TYPE)</td>
<td>DATA IN REPLY</td>
<td>DATA IN REPLY (TYPE)</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------</td>
<td>------------------------</td>
<td>--------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>51 Write dynamic assignments</td>
<td>Byte 0</td>
<td>PV transmitter variable code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1</td>
<td>SV transmitter variable code</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 2</td>
<td>TV transmitter variable code</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 3</td>
<td>FV transmitter variable code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52 Set transmitter variable zero</td>
<td>Byte 0</td>
<td>transmitter variable code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>53 Write transmitter variable units</td>
<td>Byte 0</td>
<td>transmitter variable code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1</td>
<td>transmitter variable code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>54 Read transmitter variable information</td>
<td>Byte 0</td>
<td>transmitter variable code</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1-3</td>
<td>transmitter variable code</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 4</td>
<td>transm. var. sensor serial number</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 5-8</td>
<td>transm. var. limits units code</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 9-12</td>
<td>transm. variable upper limit (F)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 13-16</td>
<td>transm. variable lower limit (F)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>transm. var. damping value (see) (F)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55 Write transmitter variable damping value</td>
<td>Byte 0</td>
<td>transmitter variable code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1-4</td>
<td>transmitter variable code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>56 Write transmitter variable sensor serial number</td>
<td>Byte 0</td>
<td>transmitter variable code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 1-3</td>
<td>transmitter variable code</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>57 Read unit tag, descriptor, date</td>
<td>none</td>
<td>none</td>
<td>Byte 0-5</td>
<td>unit tag (A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte 6-17</td>
<td>unit descriptor (A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte 18-20</td>
<td>unit date (D)</td>
</tr>
<tr>
<td>58 Write unit tag, descriptor, date</td>
<td>Byte 0-5</td>
<td>unit tag (A)</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 6-17</td>
<td>unit descriptor (A)</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte 18-20</td>
<td>unit date (D)</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>59 Write number of response preambles</td>
<td>Byte 0</td>
<td>number of response preambles</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>108 Write burst mode command number</td>
<td>Byte 0</td>
<td>burst mode command number</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>109 Burst mode control</td>
<td>Byte 0</td>
<td>burst mode control code (0 = exit, 1 = enter)</td>
<td>as in command</td>
<td></td>
</tr>
<tr>
<td>110 Read all dynamic variables</td>
<td>none</td>
<td>none</td>
<td>Byte 0</td>
<td>PV units code (F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte 1-4</td>
<td>PV value (F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte 5</td>
<td>SV units code (F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte 6-9</td>
<td>SV value (F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte 10</td>
<td>TV units code (F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte 11-14</td>
<td>TV value (F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte 15</td>
<td>FV units code (F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte 16-19</td>
<td>FV value (F)</td>
</tr>
</tbody>
</table>

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

Unmarked items are 8-, 16- or 24-bit integers.
Appendix B: HART Universal and Common Practice Commands

First byte:

**BIT 7 = 1 COMMUNICATION ERRORS:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>parity error</td>
</tr>
<tr>
<td>5</td>
<td>overrun error</td>
</tr>
<tr>
<td>4</td>
<td>framing error</td>
</tr>
<tr>
<td>3</td>
<td>checksum error</td>
</tr>
<tr>
<td>2</td>
<td>(reserved)</td>
</tr>
<tr>
<td>1</td>
<td>rx buffer overflow</td>
</tr>
<tr>
<td>0</td>
<td>(undefined)</td>
</tr>
</tbody>
</table>

Second byte:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>device malfunction</td>
</tr>
<tr>
<td>6</td>
<td>configuration changed</td>
</tr>
<tr>
<td>5</td>
<td>cold start</td>
</tr>
<tr>
<td>4</td>
<td>output current fixed</td>
</tr>
<tr>
<td>3</td>
<td>analog output saturated</td>
</tr>
<tr>
<td>2</td>
<td>variable (not primary) out of limits</td>
</tr>
<tr>
<td>1</td>
<td>primary variable out of limits</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>CODE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Update Failed&lt;br&gt;Update In Progress&lt;br&gt;Set to Nearest Possible Value</td>
</tr>
<tr>
<td>9</td>
<td>Applied Process Too High&lt;br&gt;Lower Range Value Too High&lt;br&gt;Not In Fixed Current Mode</td>
</tr>
<tr>
<td>10</td>
<td>Applied Process Too Low&lt;br&gt;Lower Range Value Too Low&lt;br&gt;MuliDrop Not Supported</td>
</tr>
<tr>
<td>11</td>
<td>In MultiDrop Mode&lt;br&gt;Invalid Transmitter Variable Code&lt;br&gt;Upper Range Value Too High</td>
</tr>
<tr>
<td>12</td>
<td>Invalid Unit Code&lt;br&gt;Upper Range Value Too Low</td>
</tr>
<tr>
<td>13</td>
<td>Both Range Values Out of Limits</td>
</tr>
<tr>
<td>14</td>
<td>Pushed Upper Range Value Over Limit&lt;br&gt;Span Too Small</td>
</tr>
</tbody>
</table>

TABLE 4-4. Command-Specific Error Codes.
Index

A
Address · 6-45
Alarm Latch · 5-11
Auto Acquisition · 6-3

B
Big-endian · 6-33
Byte Count · 6-46

C
Channel Diagnostics · 8-3
Channel HART Enable · 5-17
Checksum · 6-46
Configuring the IF4IH for the Micro 1500 · 4-2
Configuring the module for HART · 6-1
Conventions
used in the manual · vii
Copying Ladder · 4-7
Copying Routines or Programs · 4-7

D
Data · 6-46
Data Format · 5-12, 5-13
Data Formats · 1
Data Type Descriptions · 7
Demultiplexing · 6-10
Derating · 4

E
EMC Directive 2-· 1
Enable Fast Scan · 6-11
Engineering Units · 5-13
Error Codes · 8-5
European Union Directives · 1
Exporting Tags · 4-8

F
Fault Value · 5-14
Field Wiring
  General
    Terminal Block
    Grounding · 10

G
General Status Bits · 5-3
Generic Profile · 1
Generic Profile Parameters · 4
Get HART Device Information · 6-30
Grounding · 10

H
Handle Timeout · 5-17
HART Message Format · 6-44
Hart multi-drop · 6-3
HART Packet 0 · 6-5
HART Packet 1 · 6-6
HART Packet 2 · 6-7
HART Packet 3 · 6-8
HART Packet 4 · 6-8
HART Packet Data · 6-2
HART Pass-Through Command · 6-33
HART Protocol · 6-44
HART Slot Variables 0 & 1 · 5-18
HART Slot Variables 2 & 3 · 5-18
Hart Suspend · 5-20
Hazardous Location 2-· 2
High Clamp Value · 5-15
Hold Last State · 5-4

I
Idle Value
  Program Value · 5-14
Importing Tags · 4-8
Input Image · 5-2
Input Tags · 6-13

L
Latch Alarm · 5-11
LED · 2
Little-endian · 6-33
Low Clamp Value · 5-14
Low Voltage Directive 2-· 2

M
Memory Map · 5-1
Message Ladder · 6-18
Message Master Control · 5-21, 6-12
Message Request Buffer · 5-21, 6-13
Message Request Size · 5-21, 6-13
Message Response Buffer · 5-5, 6-14  
Message Response Size · 5-5, 6-14  
Message Slave Control · 5-5, 6-14  
Messages · 6-12  
Micro 1500 Ladder Samples · 4-6  
MicroLogix 1500 Addressing · 4-2  
Minimum spacing 2 · 5  
Module Configuration · 5-5  
Module Error · 8-3  
Module Operation · 3  
module specific commands · 6-3  
Module Specific Commands · 6-30  
Module Update Time · 5-21  

**N**

Noise  
Reducing · 4

**O**

Out of Service Status Bits · 5-3  
Output Data File · 5-18  
Output Data Values · 5-4  
Output Tags · 6-12  
Output Type · 5-12, 5-13  
output types · 1  
Over-Range Flag Bits · 5-4

**P**

Packed ASCII · 7-3  
Packet Interval · 6-11  
Packet Just Scanned/Fast Scan · 5-20  
Paging Error Codes · 6-14  
Panel Mounting · 8  
Pass-Through Ladder · 6-38  
Pass-Through Scheme · 5-17  
Percent Range · 5-13  
Power Requirements · 2  
Power Supply Distance · 4  
Power-up Diagnostics · 8-2  
Preamble · 6-44  
Processing a Message · 6-15  
Program Value  
Idle Value · 5-14

**R**

Ramp Rate · 5-15  
**Raw/Proportional Data** · 5-13  
Receiving Message · 6-17  
Reset/Reconfig · 7-1

**S**

Safety Considerations · 8-1  
Sample Ladder · 10  
Scaled-for-PID · 5-13  
Sending a HART Command · 6-46  
Sending Message · 6-16  
Specifications · 1  
Start Character · 6-44  
Status · 5-3, 6-46  
Swap Byte Order · 7-3  
System Operation · 2

**T**

Tags · 7  
Technical support  
contact information · vii  
Terminal Block · 10

**U**

Under-Range Flag Bits · 5-4  
Unlatch Process High Alarm · 5-19  
Unlatch Process Low Alarm · 5-19  
User Defined Data Types · 6

**W**

Wire Size · 12  
Wiring Diagram · 13