029747RA061
20 August, 1997
Version 1.2

Specification for

Modbus Interface to Excalibur 7000

Robertshaw
INDUSTRIAL PRODUCTS

1602 Mustang Drive
Maryville, Tennessee 37801
Phone: (865) 981-3100 Fax: (865) 981-3100

Table of Contents

1. The Modbus Protocol 1

1.1 Introduction To Modbus 1

1.2 Serial Data Transmission 2

1.3 Modbus Message Framing 2

1.3.1 Recognizing the Beginning and End of a Frame 2

1.3.2 The Address Field 3

1.3.3 The Function Field 3

1.3.4 The Data Field 4

1.3.5 The Error Check Field 4

1.4 Exception Responses 5

1.4.1 Reply Construction 5

1.4.2 Exception Codes 6

2. Excalibur Set Up 7

2.1 Communication Option 7

2.2 Communication Parameters 7

2.3 Modbus Device Address 7

3. Supported Function Codes 8

3.1 Summary 8

3.2 01 - Read Coil Status 9

3.2.1 Description 9

3.2.2 Query 9

3.2.3 Response 9

3.3 02 - Read Input Status 10

3.3.1 Description 10

3.3.2 Query 10

3.3.3 Response 10

3.4 03 - Read Holding Registers 11

3.4.1 Description 11

3.4.2 Query 11

3.4.3 Response 11

3.5 04 - Read Input Registers 12

3.5.1 Description 12

3.5.2 Query 12

3.5.3 Response 12

3.6 05 - Force Single Coil 13

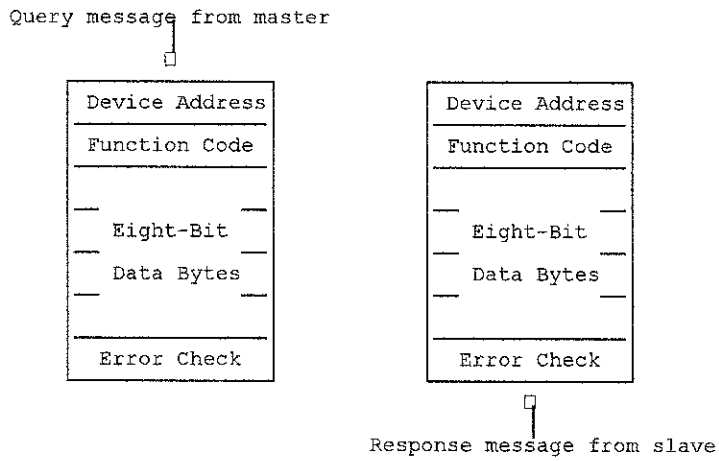
3.6.1 Description....13
3.6.2 Query....13
3.6.3 Response....13
3.7 06 - Preset Single Register....14
3.7.1 Description....14
3.7.2 Query....14
3.7.3 Response....14
3.8 15 - Force Multiple Coils....15
3.8.1 Description....15
3.8.2 Query....15
3.8.3 Response....15
3.9 16 - Preset Multiple Registers....16
3.9.1 Description....16
3.9.2 Query....16
3.9.3 Response....16
4. Register Address Map....17
4.1 Address Ranges....17
4.2 Data Types....18
4.3 Coil Status (Read/Write Discreet)....19
4.4 Input Status (Read Only Discreet)....20
4.5 Holding Registers (Read/Write)....21
4.6 Input Registers (Read Only)....24
4.7 Unit Code Table....25

1. The Modbus Protocol

1.1 Introduction To Modbus

The Modbus protocol is widely used in a number of process industry products including instruments, distributed control systems, remote terminal units, multiplexers, and data acquisition equipment.

The Modbus protocol uses a master-slave technique, in which only one device (the master) can initiate transactions (called queries). The other devices (the slaves) respond by supplying the requested data to the master, or by taking the action requested in the query. The Excalibur is a Modbus slave device. Thus all communication follows a query response cycle as illustrated below.



The Query: The function code in the query tells the addressed slave device what kind of action to perform. The data bytes contain any additional information that the slave will need to perform the function. For example, function code 03 will query the slave to read holding registers and respond with their contents. The data field must contain the information telling the slave which register to start at and how many registers to read. The error check field provides a method for the slave to validate the integrity of the message contents.

The Response: If the slave makes a normal response, the function code in the response is an echo of the function code in the query. The data bytes contain the data collected by the slave, such as register values or status. If an error occurs, the function code is modified to indicate that the response is an error response, and the data bytes contain a code that

describes the error. The error check field allows the master to confirm that the message contents are valid.

1.2 Serial Data Transmission

The Modbus specification defines two types of serial transmission: ASCII and RTU (Remote Terminal Unit). The Excalibur, however, only supports the RTU mode. The RTU Mode provides better data throughput than ASCII for the same baud rate.

1.3 Modbus Message Framing

The Modbus message is placed by the transmitting device into a frame that has a known beginning and ending point. This allows receiving devices to begin at the start of the message, read the address portion and determine which device is addressed, and to know when the message is completed. Partial messages can be detected and errors can be set as a result.

1.3.1 Recognizing the Beginning and End of a Frame

In RTU mode, messages start with a silent interval of at least 3.5 character times. This is most easily implemented as a multiple of character times at the baud rate that is being used on the network (shown as T1-T2-T3-T4 in the figure below).

The first field then transmitted is the device address.

Networked devices monitor the network bus continuously, including during the 'silent' intervals. When the first field (the address field) is received, each device decodes it to find out if it is the addressed device.

Following the last transmitted character, a similar interval of at least 3.5 character times marks the end of the message. A new message can begin after this interval.

The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame, the receiving device flushes the incomplete message and assumes that the next byte will be the address field of a new message.

Similarly, if a new message begins earlier than 3.5 character times following a previous message, the receiving device will consider it a continuation of the previous message. This will set an error, as the value in the final CRC field will not be valid for the combined messages. A typical message frame is shown below.

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1-T2-T3-T4	8 Bits	8 Bits	n x 8 Bits	16 Bits	T1-T2-T3-T4

The master is typically configured by the user to wait for a predetermined timeout interval before aborting the transaction. This interval is set to be long enough for any slave to respond normally. If the slave detects a transmission error, the message will not be acted upon. The slave will not construct a response to the master. The timeout will expire allowing the master to handle the error. Note that a message addressed to a nonexistent slave device will also cause a timeout.

If parity is enabled, then both the master and the slave check each transmitted byte for parity error. Any byte received with an error invalidates the entire message.

1.3.2 The Address Field

The first byte of a message is the address field. Valid slave device addresses are in the range of 0 - 247 decimal. The individual slave devices are assigned addresses in the range of 1 - 247. A master addresses a slave by placing the slave address in the address field of the message. When the slave sends its response, it places its own address in the address field of the response to let the master know which slave is responding.

Address 0 is used for the broadcast address. Broadcast messages are more appropriate for PLC (Programmable Logic Controller) devices than for instruments, so the Excalibur does not support the broadcast address.

1.3.3 The Function Field

The second byte of a message is the function code field. Valid codes are in the range of 1 to 127 decimal. The Excalibur supports only a limited set of these function codes.

When a message is sent from a master to a slave device the function code field tells the slave what kind of action to perform.

When the slave responds to the master, it uses the function code field to indicate either a normal (error-free) response or that some kind of error occurred (called an exception response). For a normal response, the slave simply echoes the original function code. For an exception response, the slave returns a code that is equivalent to the original function code with its most-significant bit set to 1.

For example, a message from master to slave to read a group of holding registers would have the following function code:

```
0000   0011   (Hexadecimal  03)
```

If the slave device takes the requested action without error, it returns the same code in its response. If an exception occurs, it returns:

```
1000   0011   (Hexadecimal 83)
```

In addition to its modification of the function code for an exception response, the slave places a unique code into the data field of the response message. This tells the master what kind of error occurred, or the reason for the exception. The master device's application program has the responsibility of handling exception responses.

1.3.4 The Data Field

The data field of messages sent from a master to a slave device will contain one or more bytes of additional information that the slave must use to take the action defined by the function code. This can include items like addresses, the quantity of items to be handled, and the count of actual data bytes in the field.

For example, if the master requests a slave to read a group of holding registers (function code 03), the data field specifies the starting register and how many registers are to be read. If the master writes to a group of registers in the slave (function code 16), the data field specifies the starting register, how many registers to write, the count of data bytes to follow in the data field, and the data to be written into the registers.

If no error occurs, the data field of a response from a slave to a master contains the data requested. If an error occurs, the field contains an exception code that the master application can use to determine the next action to be taken.

1.3.5 The Error Check Field

The error check field contains a 16-bit value implemented as two 8-bit bytes. The error check value is the result of a Cyclical Redundancy Check calculation performed on the entire message contents.

The CRC field is appended to the message as the last field in the message. When this is done, the low-order byte of the field is appended first, followed by the high-order byte. The CRC high-order byte is the last byte to be sent in the message.

The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, the message is invalid.

The CRC is started by first pre-loading a 16-bit register to all 1's. Then a process begins of applying successive 8-bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit if one is used, do not apply to the CRC.

During generation of the CRC, each 8-bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined, If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8-bit byte is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final content of the register, after all the bytes of the message have been applied, is the CRC value.

A procedure for generating a CRC is:

1. Load a 16-bit register with FFFF hex (all 1's). Call this the CRC register.
2. Exclusive OR the first 8-bit byte of the message with the low-order byte of the 16-bit CRC register, putting the result in the CRC register.
3. Shift the CRC register one bit to the right (toward the LSB), zero-filling the MSB. Extract and examine the LSB.
4. (If the LSB was 0): Repeat Step 3 (another shift).
   (If the LSB was 1): Exclusive OR the CRC register with the polynomial value A001.
5. Repeat Steps 3 and 4 until 8 shifts have been performed. When this is done, a complete 8-bit byte will have been processed.
6. Repeat Steps 2 through 5 for the next 8-bit byte of the message. Continue doing this until all bytes have been processed.
7. The final content of the CRC register is the CRC value.

1.4 Exception Responses

1.4.1 Reply Construction

When a master device sends a query to a slave device it expects a normal response. One of four possible events can occur from the master's query:

1. If the slave device receives the query without a communication error, and can handle the query normally, it returns a normal response.
2. If the slave does not receive the query due to a communication error, no response is returned. The master program will eventually process a timeout condition for the query.
3. If the slave receives the query, but it detects a communication error (parity or CRC), no response is returned. The master program will eventually process a timeout condition for the query.

4. If the slave receives the query without a communication error, but it cannot handle it (for example, if the request is to read a non-existent address), the slave will return an exception response informing the master of the nature of the error.

The exception response message has two fields that differentiate it from a normal response.

Function Code Field

In a normal response, the slave echoes the function code of the original query in the function code field of the response. All function codes have a most-significant bit(MSB) of 0 (their values are all below 80 hexadecimal). In an exception response, the slave sets the MSB of the function code to 1. This makes the function code value in an exception response exactly 80 hexadecimal higher than the value would be for a normal response.

With the function code's MSB set, the master's application program can recognize the exception response and can examine the data field for the exception code.

Data Field

In a normal response, the slave may return data or statistics in the data field (any information that was requested in the query). In an exception response, the slave returns an exception code in the data field. This defines the slave condition that caused the exception.

1.4.2 Exception Codes

Code	Name	Meaning
01	Illegal Function	The function code received in the query is not an allowable action for the slave.
02	Illegal Data Address	The data address received in the query is not an allowable address for the slave.
03	Illegal Data Value	A value contained in the query data field is not an allowable value for the slave.
04	Slave Device Failure	An unrecoverable error occurred while the slave was attempting to perform the requested action.
06	Slave Device Busy	The slave is engaged in processing a long-duration program command. The master should retransmit the message later when the slave is free.

2. Excalibur Set Up

2.1 Communication Option

The Excalibur supports the Modbus protocol RTU mode with either the RS232 or the RS485 option board. With RS232, only one slave device can be connected to the master. With RS485, multiple slave devices can be connected to the same master.

2.2 Communication Parameters

When setting up a network of devices that will communicate with the Modbus protocol, each device on the link must be configured with the same communication parameters. Ensure that each device and the host are set for the same data rate, word size, parity, start, and stop bits. These parameters can only be set via the Excalibur front panel.

2.3 Modbus Device Address

To accommodate multiple devices on a single Modbus link, each device on a link must have a unique Modbus address. Valid addresses range from 1 to 247. This can only be set via the Excalibur front panel.

3. Supported Function Codes

3.1 Summary

To ensure the widest compatibility with these products, the Excalibur level transmitter supports only fundamental Modbus commands. These are enumerated in the following table.

Supported Modbus Commands

Command	Name	Description
1	Read Coil Status	Reads the status of coils, essentially discreet output (on/off) points.
2	Read Input Status	Reads the state of a discreet inputs (on/off).
3	Read Holding Registers	Reads one or more holding registers, each of which is 16 bits wide. Multiple sequential registers are used to construct floating point values and strings. Holding registers are for output values, i.e., items that can be changed under program control. Excalibur variables such as setpoints, ranges, and tag are mapped to holding registers.
4	Read Input Registers	Reads one or more input registers, each of which is 16 bits wide. Multiple sequential registers are used to construct floating point values. Input registers are for read-only items. Excalibur variables such as level, volume, and flow are mapped to input registers.
5	Force Single Coil	Writes a single coil value.
6	Preset Single Holding Register	Writes a single holding register value.
15	Force Multiple Coils	Writes multiple consecutive coil values.
16	Preset Multiple Holding Registers	Writes multiple consecutive holding registers.

3.2 01 - Read Coil Status

3.2.1 Description

Reads the ON/OFF status of discrete outputs in the slave.

3.2.2 Query

The query message specifies the starting coil and quantity of coils to be read. Coils are addressed starting at 0000. The following example shows a request to read coils 0 to 9 from slave address 17 (Hex 11).

Field Name	**(Hex)**
Slave Address	11
Function	01
Starting Address Hi	00
Starting Address Low	00
No. of Points Hi	00
No. of Points Low	0A
Error Check (CRC)	

3.2.3 Response

The coil status in the response message is packed as one coil per bit of the data field. Status is indicated as: 1 = ON; 0 = OFF. The LSB of the first data byte contains the coil addressed in the query. The other coils follow toward the high order end of this byte, and from 'low order to high order' in subsequent bytes.

If the returned coil quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros (toward the high order end of the byte). The Byte Count field specifies the quantity of complete bytes of data. Here is the example response to the query.

Field Name	**(Hex)**
Slave Address	11
Function	01
Byte Count	02
Data (Coils 0000 - 0007)	FF
Data (Coils 0008 - 0009)	03
Error Check (CRC)	

3.3 02 - Read Input Status

3.3.1 Description

Reads the ON/OFF status of discrete inputs in the slave.

3.3.2 Query

The query message specifies the starting input and quantity of inputs to be read. Inputs are addressed starting at 0000. Here is an example of a request to read all eight of the defined inputs for the Excalibur (Inputs 0 to 7) at slave address 17:

Field Name	**(Hex)**
Slave Address	11
Function	02
Starting Address Hi	00
Starting Address Low	00
No. of Points Hi	00
No. of Points Low	08
Error Check (CRC)	

3.3.3 Response

The input status in the response message is packed as one input per bit of the data field. Status is indicated as: 1 = ON; 0 = OFF. The LSB of the first data byte contains the input addressed in the query. The other inputs follow toward the high order end of this byte, and from 'low order to high order' in subsequent bytes.

If the returned input quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros (toward the high order end of the byte). The Byte Count field specifies the quantity of complete bytes of data.

Here is the example response to the query when all requested values are on:

Field Name	**(Hex)**
Slave Address	11
Function	02
Byte Count	01
Data (Inputs 0000 - 0007)	FF
Error Check (CRC)	

3.4 03 - Read Holding Registers

3.4.1 Description

Reads the binary contents of holding registers in the slave. In the Excalibur, the starting address must correspond to a mapped holding register starting address, and the length must include the entire variable. No partial variables will be returned. Any request for a partial variable (e.g. only 1 register of a floating point value) will return exception code 2 - Illegal Data Address.

3.4.2 Query

The query message specifies the starting register and quantity of registers to be read. Registers are addressed starting at 0000. Here is an example of a request to read registers 0012 and 0013 (the Process Alarm 2 Low Setpoint) from slave address 17.

Field Name	**(Hex)**
Slave Address	11
Function	03
Starting Address Hi	00
Starting Address Low	0C
No. of Points Hi	00
No. of Points Low	02
Error Check (CRC)	

3.4.3 Response

The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits.

Here is the example response to the query when the setpoint is 10.0.

Field Name	**(Hex)**
Slave Address	11
Function	03
Byte Count	04
Data Hi (Register 0002)	41
Data Low (Register 0002)	20
Data Hi (Register 0003)	00
Data Low (Register 0003)	00
Error Check (CRC)	

3.5 04 - Read Input Registers

3.5.1 Description

Reads the binary contents of input registers in the slave. In the Excalibur, the starting address must correspond to a mapped input register starting address, and the length must include the entire variable. No partial variables will be returned. Any request for a partial variable (e.g. only 1 register of a floating point value) will return exception code 2 - Illegal Data Address.

3.5.2 Query

The query message specifies the starting register and quantity of registers to be read. Registers are addressed starting at 0000. Here is an example of a request to read the volume (registers 2 and 3) from slave address 17:

Field Name	**(Hex)**
Slave Address	11
Function	04
Starting Address Hi	00
Starting Address Low	02
No. of Points Hi	00
No. of Points Low	02
Error Check (CRC)	

3.5.3 Response

The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits.

Here is the example response to the query when the volume is not valid (not a number).

Field Name	**(Hex)**
Slave Address	11
Function	04
Byte Count	04
Data Hi (Register 0002)	7F
Data Low (Register 0002)	A0
Data Hi (Register 0003)	00
Data Low (Register 0003)	00
Error Check (CRC)	

3.6 05 - Force Single Coil

3.6.1 Description

Forces a single coil to either ON or OFF.

3.6.2 Query

The query message specifies the coil address to be forced. Coils are addressed starting at 0000.

The requested ON/OFF state is specified by a constant in the query data field. A value of FF 00 hex requests the coil to be ON. A value of 00 00 requests it be OFF. All other values are illegal and will not affect the coil. Here is an example of a request to enable Process Alarm 3 (force coil 2 ON) in slave address 17:

Field Name	**(Hex)**
Slave Address	11
Function	05
Coil Address Hi	00
Coil Address Low	02
Force Data Hi	FF
Force Data Low	00
Error Check (CRC)	

3.6.3 Response

The normal response is an echo of the query, returned after the coil state has been forced.

Here is the example response to the query:

Field Name	**(Hex)**
Slave Address	11
Function	05
Coil Address Hi	00
Coil Address Low	02
Force Data Hi	FF
Force Data Low	00
Error Check (CRC)	

3.7 06 - Preset Single Register

3.7.1 Description

Writes a value into a single holding register. In the Excalibur, this command only works for addresses that are part of a Word length variable. For any other data type, the response will contain exception code 2 - Illegal Data Address.

3.7.2 Query

The query message specifies the register reference to be written. Registers are addressed starting at 0000.

This example sets the Process Alarm 1 relay (address 0008) to 2.

Field Name	(Hex)
Slave Address	11
Function	06
Holding Register Address Hi	00
Holding Register Address Low	08
Data Hi	00
Data Low	02
Error Check (CRC)	

3.7.3 Response

The normal response is an echo of the query, returned after the register contents have been changed.

Field Name	(Hex)
Slave Address	11
Function	06
Holding Register Address Hi	00
Holding Register Address Low	08
Data Hi	00
Data Low	02
Error Check (CRC)	

3.8 15 - Force Multiple Coils

3.8.1 Description

Forces each coil in a sequence of coils to either ON or OFF.

3.8.2 Query

The query message specifies the coil references to be forced. Coils are addressed starting at 0000. The requested ON/OFF states are specified by contents of the query data field. A logical '1' in a bit position of the field requests the corresponding coil to be ON. A logical '0' requests it to be OFF.

This example sets all 10 coils in the Excalibur. It turns on the process alarms, while turning off the setpoint and output alarms. PID is set to on, but in manual mode.

Field Name	**(Hex)**
Slave Address	11
Function	0F
Coil Address Hi	00
Coil Address Low	00
Quantity of Coils Hi	00
Quantity of Coils Low	0A
Byte Count	02
Force Data (coils 0000 to 0007)	0F
Force Data (coils 0008 - 0009)	01
Error Check (CRC)	

3.8.3 Response

The normal response returns the slave address, function code, starting address, and quantity of coils forced.

Field Name	**(Hex)**
Slave Address	11
Function	0F
Coil Address Hi	00
Coil Address Low	00
Quantity of Coils Hi	00
Quantity of Coils Low	0A
Error Check (CRC)	

3.9 16 - Preset Multiple Registers

3.9.1 Description

Writes values into a sequence of holding registers. The starting address must correspond to a mapped holding register starting address, and the length must include the entire variable. No partial variables will be written. Any write to a partial variable (e.g. only 1 register of a floating point value) will return exception code 2 - Illegal Data Address.

3.9.2 Query

The query message specifies the register references to be preset. Registers are addressed starting at 0000. Data is packed as two bytes per register.

Field Name	**(Hex)**
Slave Address	11
Function	10
Starting Holding Register Address Hi	00
Starting Holding Register Address Low	00
Number of Registers Hi	00
Number of Registers Low	02
Byte Count	04
Data	00
Data	00
Data	00
Data	00
Error Check (CRC)	

3.9.3 Response

The normal response returns the slave address, function code, starting address, and quantity of registers preset.

Field Name	**(Hex)**
Slave Address	11
Function	10
Starting Holding Register Address Hi	00
Starting Holding Register Address Low	00
Number of Registers Hi	00
Number of Registers Low	02
Error Check (CRC)	

4. Register Address Map

4.1 Address Ranges

The tables in this section define the register addresses for the Excalibur variables that are accessible via the Modbus port. Not all Excalibur variables are accessible. Many of the configuration parameters (such as the selection of level versus flow versus volume operating mode) may be read via Modbus commands, but can only be changed via the Excalibur front panel.

The original specification for Modbus addresses assumed that the slave device was a Programmable Logic Controller (PLC). The various types of inputs and outputs were mapped to PLC addresses as follows:

Type	Old Address Ranges	Modbus Commands
Coils (Discreet Outputs)	1-10000	1, 5, 15
Inputs (Discreet)	10001-20000	2
Input Registers	30001-40000	4
Holding Registers (Outputs)	40001-50000	3, 6, 16

These address ranges were specific to the Gould-Modicon family of PLC's. However, the commands used to access these addresses all use a zero relative addressing scheme. Thus PLC inputs 10001 to 20000 were addressed in command 2 as 0 to 9999.

All of the addresses in this section are zero relative addresses, just like they are used in the commands. If your Modbus master requires the old style addresses, simply add the appropriate starting address to the table value.

In these tables, all address values are in decimal.

4.2 Data Types

In the address map tables, a data type is shown for each input register and holding register variable. The data type determines the number of consecutive registers that comprise the complete variable. When accessing these variables using commands 3, 4, 6, and 16, the starting address must correspond to the beginning address of a variable, and the length must be correct for the data type. If multiple values are read simultaneously, the total length must encompass all assigned registers for the contiguous variables.

Data Type	Number of Registers	Description
Float	2	IEEE-754 format (4 bytes)
Word	1	Unsigned 16 bit quantity
String[x]	x/2	ASCII string, byte length is always a multiple of 2

4.3 Coil Status (Read/Write Discreet)

Coil addresses represent items that are discreet in nature (on/off only) and that can be read and written via Modbus commands.

To determine the old style Modbus address from the table value, just add 1.

Address	Variable	Description
0000	Process Alarm 1 Status	0 = Disabled, 1 = Enabled
0001	Process Alarm 2 Status	0 = Disabled, 1 = Enabled
0002	Process Alarm 3 Status	0 = Disabled, 1 = Enabled
0003	Process Alarm 4 Status	0 = Disabled, 1 = Enabled
0004	Setpoint Alarm 1 Status	0 = Disabled, 1 = Enabled
0005	Setpoint Alarm 2 Status	0 = Disabled, 1 = Enabled
0006	Output Alarm 1 Status	0 = Disabled, 1 = Enabled
0007	Output Alarm 2 Status	0 = Disabled, 1 = Enabled
0008	PID State	PID controller, 0 = Off, 1 = On
0009	PID Control	Auto/Manual control. 0 = Manual, 1 = Auto
0010	Reset Config Changed.	This always read 0. Write 1 to clear the configuration changed bit (input 0006).

4.4 Input Status (Read Only Discreet)

Input addresses represent items that are discreet in nature (on/off only) and that can only be read via Modbus commands.

To determine the old style Modbus address from the table value, just add 10,001.

Address	Variable	Description
0000	PV Out of Limits	These first 8 bits are the equivalent of the HART Hardware Status byte. 0 = OK
0001	Non-PV Out of Limits	
0002	PV Analog Output Saturated	
0003	PV Analog Output Fixed	
0004	More Status Available	
0005	Cold Start	
0006	Configuration Changed	
0007	Device Malfunction	
0008	RESERVED	always zero
0009	RESERVED	always zero
0010	RESERVED	always zero
0011	Calibration Failure	0 = OK, 1 = Error
0012	PFM Failure	0 = OK, 1 = Error
0013	RAM Test Failure	0 = OK, 1 = Error
0014	ROM Checksum Error	0 = OK, 1 = Error
0015	RESERVED	always zero
0016	Process Alarm 1	0 = No Alarm, 1 = Alarm tripped
0017	Process Alarm 2	0 = No Alarm, 1 = Alarm tripped
0018	Process Alarm 3	0 = No Alarm, 1 = Alarm tripped
0019	Process Alarm 4	0 = No Alarm, 1 = Alarm tripped
0020	Setpoint Alarm 1	0 = No Alarm, 1 = Alarm tripped
0021	Setpoint Alarm 2	0 = No Alarm, 1 = Alarm tripped
0022	Output Alarm 1	0 = No Alarm, 1 = Alarm tripped
0023	Output Alarm 2	0 = No Alarm, 1 = Alarm tripped

4.5 Holding Registers (Read/Write)

Holding Register addresses represent items that are not discreet (e.g., floating point, strings, etc.) and that can be read and written via Modbus commands. Since each register is 16 bits (2 bytes) wide, multiple sequential registers are used to construct floating point values and strings.

To determine the old style Modbus address from the table value, just add 40001.

Address	Type	Variable	Description
0000	Float	Process alarm 1 LSP	Low setpoint (%)
0002	Float	Process alarm 1 HSP	High setpoint (%)
0004	Float	Process alarm 1 TD On	Time delay (seconds) on alarm
0006	Float	Process alarm 1 TD Off	Time delay (seconds) on clear
0008	Word	Process alarm 1 Relay	Relay to operate. 0 = no relay. Otherwise 1 to 4.
0009	Word	Process alarm 1 Source	0 = Level, 1 = Volume, or 2 = Flow
0010	Word	Process alarm 1 Failsafe	Failsafe mode. 0 = Low, 1 = High
0011	Word	Process alarm 1 Type	1 = Fixed, 2 = Adjustable differential
0012	Float	Process alarm 2 LSP	Low setpoint (%)
0014	Float	Process alarm 2 HSP	High setpoint (%)
0016	Float	Process alarm 2 TD On	Time delay (seconds) on alarm
0018	Float	Process alarm 2 TD Off	Time delay (seconds) on clear
0020	Word	Process alarm 2 Relay	Relay to operate. 0 = no relay. Otherwise 1 to 4.
0021	Word	Process alarm 2 Source	0 = Level, 1 = Volume, or 2 = Flow
0022	Word	Process alarm 2 Failsafe	Failsafe mode. 0 = Low, 1 = High
0023	Word	Process alarm 2 Type	1 = Fixed, 2 = Adjustable differential
0024	Float	Process alarm 3 LSP	Low setpoint (%)
0026	Float	Process alarm 3 HSP	High setpoint (%)
0028	Float	Process alarm 3 TD On	Time delay (seconds) on alarm
0030	Float	Process alarm 3 TD Off	Time delay (seconds) on clear
0032	Word	Process alarm 3 Relay	Relay to operate. 0 = no relay. Otherwise 1 to 4.
0033	Word	Process alarm 3 Source	0 = Level, 1 = Volume, or 2 = Flow
0034	Word	Process alarm 3 Failsafe	Failsafe mode. 0 = Low, 1 = High
0035	Word	Process alarm 3 Type	1 = Fixed, 2 = Adjustable differential
0036	Float	Process alarm 4 LSP	Low setpoint (%)
0038	Float	Process alarm 4 HSP	High setpoint (%)
0040	Float	Process alarm 4 TD On	Time delay (seconds) on alarm
0042	Float	Process alarm 4 TD Off	Time delay (seconds) on clear
0044	Word	Process alarm 4 Relay	Relay to operate. 0 = no relay. Otherwise 1 to 4.
0045	Word	Process alarm 4 Source	0 = Level, 1 = Volume, or 2 = Flow

Address	Type	Variable	Description
0046	Word	Process alarm 4 Failsafe	Failsafe mode. 0 = Low, 1 = High
0047	Word	Process alarm 4 Type	1 = Fixed, 2 = Adjustable differential
0048	Float	Setpoint alarm 1 LSP	Low setpoint (%)
0050	Float	Setpoint alarm 1 HSP	High setpoint (%)
0052	Word	Setpoint alarm 1 Relay	Relay to operate. 0 = no relay. Otherwise 1 to 4.
0053	Word	Setpoint alarm 1 Failsafe	Failsafe mode. 0 = Low, 1 = High
0054	Word	Setpoint alarm 1 Type	1 = Fixed, 2 = Adjustable differential
0055	Float	Setpoint alarm 2 LSP	Low setpoint (%)
0057	Float	Setpoint alarm 2 HSP	High setpoint (%)
0059	Word	Setpoint alarm 2 Relay	Relay to operate. 0 = no relay. Otherwise 1 to 4.
0060	Word	Setpoint alarm 2 Failsafe	Failsafe mode. 0 = Low, 1 = High
0061	Word	Setpoint alarm 2 Type	1 = Fixed, 2 = Adjustable differential
0062	Float	Output alarm 1 LSP	Low setpoint (%)
0064	Float	Output alarm 1 HSP	High setpoint (%)
0066	Word	Output alarm 1 Relay	Relay to operate. 0 = no relay. Otherwise 1 to 4.
0067	Word	Output alarm 1 Failsafe	Failsafe mode. 0 = Low, 1 = High
0068	Word	Output alarm 1 Type	1 = Fixed, 2 = Adjustable differential
0069	Float	Output alarm 2 LSP	Low setpoint (%)
0071	Float	Output alarm 2 HSP	High setpoint (%)
0073	Word	Output alarm 2 Relay	Relay to operate. 0 = no relay. Otherwise 1 to 4.
0074	Word	Output alarm 2 Failsafe	Failsafe mode. 0 = Low, 1 = High
0075	Word	Output alarm 2 Type	1 = Fixed, 2 = Adjustable differential
0076	Float	Level URV	Level Upper Range Value. This is the 20 mA point when level is controlling the analog output. Units are the same as the Level Reading.
0078	Float	Level LRV	Level Lower Range Value. This is the 4 mA point when level is controlling the analog output.
0080	Float	Level Damping	Damping value (in seconds)
0082	Float	Volume URV	Volume Upper Range Value. This is the 20 mA point when volume is controlling the analog output.
0084	Float	Volume LRV	Volume Lower Range Value. This is the 4 mA point when volume is controlling the analog output.
0086	Float	Flow URV	Flow Upper Range Value. This is the 20 mA point when Flow is controlling the analog output.
0088	Float	Flow LRV	Flow Lower Range Value. This is the 4 mA point when Flow is controlling the analog output.
0090	Float	PID Setpoint	Units are the same as the input source (level, volume, or flow)
0092	Float	PID Gain	Value is in %

Address	Type	Variable	Description
0094	Float	PID Reset Time	Value is in repeats per minute
0096	Float	PID Rate Time	Value is in minutes
0098	Float	PID Manual setting	Output value for the PID when it is in manual mode. If not in manual, writes to this address have no effect. Units are percent.
0100	Word	PID Auto-Tune Mode	0 = Off, 1 = Mild, 2 = Standard, 3 = Aggressive, 4 = Fastest.
0101	Word	PID Source	Variable providing the input to the PID calculation. 0 = Level, 1 = Volume, 2 = Flow
0102	String[8]	Tag	This is an ASCII field that can be used to store and retrieve a tag for the instrument. This uses 4 consecutive registers to hold the 8 bytes.
0106	String[16]	Description	This is an ASCII field that can be used to store and retrieve a description for the instrument. This uses 8 consecutive registers to hold the 16 bytes.

4.6 Input Registers (Read Only)

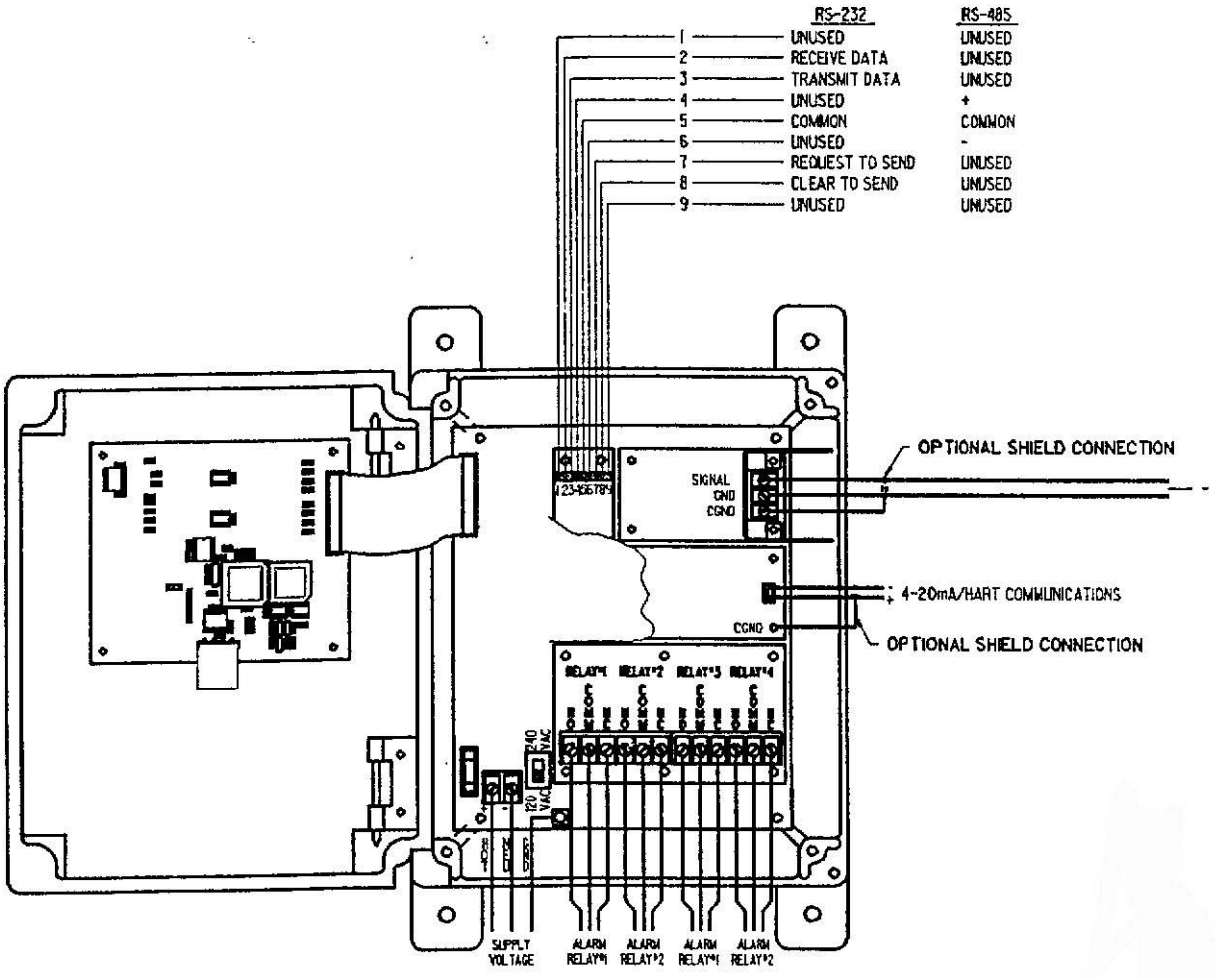
Input Register addresses represent items that are not discreet (e.g., floating point, strings, etc.) and that can only be read via Modbus commands. Since each register is 16 bits (2 bytes) wide, multiple sequential registers are used to construct floating point values and strings.

To determine the old style Modbus address from the table value, just add 30001.

Address	Type	Variable	Description
0000	Float	Level Reading	Present level reading. Always available.
0002	Float	Volume Reading	Present volume reading. Only valid when the Measurement Mode is Volume. Otherwise, this value is set to Not A Number (7F A0 00 00).
0004	Float	Flow Reading	Present flow reading. Only valid when the Measurement Mode is Flow. Otherwise, this value is set to Not A Number (7F A0 00 00).
0006	Float	PID Output	The output of the PID algorithm (%).
0008	Float	Loop Current	The present value of the loop current in milliamps.
0010	Word	Level Units	See the Unit Code table for interpretation.
0011	Word	Volume Units	See the Unit Code table for interpretation.
0012	Word	Flow Units	See the Unit Code table for interpretation.
0013	Word	Measurement Mode	0 = Level, 1 = Volume, 2 = Flow.

4.7 Unit Code Table

Code	Unit Description
15	cubic ft/min
16	gallons/min
17	liters/min
18	imperial gallons/min
19	cubic meters/hr
22	gallons/sec
23	mil gallons/day
24	liters/sec
25	mil liters/day
26	cubic ft/sec
28	cubic meters/sec
30	imperial gallons/hr
40	gallons
41	liters
42	imperial gallons
43	cubic meters
44	feet
45	meter
46	barrels
47	inches
48	cm
49	mm
111	cubic yards
112	cubic feet
130	cubic ft/hr
131	cubic meters/min
136	gallons/hr
137	imperial gallons/sec
138	liters/hr
240	mil imperial gallons/day


RS-232
RS-485
1 UNUSED UNUSED
2 RECEIVE DATA UNUSED
3 TRANSMIT DATA UNUSED
4 UNUSED +
5 COMMON COMMON
6 UNUSED -
7 REQUEST TO SEND UNUSED
8 CLEAR TO SEND UNUSED
9 UNUSED UNUSED
OPTIONAL SHIELD CONNECTION
SIGNAL
CND
CGND
4-20mA/HART COMMUNICATIONS
CGND
OPTIONAL SHIELD CONNECTION
RELAY#1
RELAY#2
RELAY#3
RELAY#4
240 VAC
120 VAC
SUPPLY VOLTAGE
ALARM RELAY#1
ALARM RELAY#2
ALARM RELAY#1
ALARM RELAY#2

U.S.A and Canada
Robertshaw Industrial Products
1602 Mustang Drive
Maryville, Tennessee 37801
Telephone: (865) 981-3100 Fax: (865) 981-3168
http://www.robertshaw.thomasregister.com
http://www.robertshawindustrial.com

Exports
Invensys Appliance Controls
1701 Byrd Avenue
P.O. Box 26544
Richmond, Virginia 23261-6544
Telephone: (804) 756-6500 Fax: (804) 756-6561

invensys®

Printed in U.S.A.