WebMaster® ONE

Modbus TCP/IP Option
Instruction Manual

s800v004, s801v003 and lower

WALCHEM

Notice

Five Boynton Road, Holliston, MA 01746 USA

(508) 429-1110

Printed in USA

Proprietary Material

Statement of Limited Warranty

WALCHEM Corporation warrants equipment of its manufacture, and bearing its identification to be free from defects in workmanship and material for a period of 24 months for electronics and 12 months for mechanical parts from date of delivery from the factory or authorized distributor under normal use and service and otherwise when such equipment is used in accordance with instructions furnished by WALCHEM Corporation and for the purposes disclosed in writing a the time of purchase, if any. WALCHEM Corporation's liability under this warranty shall be limited to replacement or repair, F.O.B. Holliston, MA U.S.A. of any defective equipment or part which, having been returned to WALCHEM Corporation, transportation charges prepaid, has been inspected and determined by WALCHEM Corporation to be defective. Replacement elastomeric parts and glass components are expendable and are not covered by any warranty.

THIS WARRANTY IS IN LIEU OF ANY OTHER WARRANTY, EITHER EXPRESS OR IMPLIED, AS TO DESCRIPTION, QUALITY, MERCHANTABILITY, and FITNESS FOR ANY PARTICULAR PURPOSE OR USE, OR ANY OTHER MATTER.

180277 Rev B

February 2006

TABLE OF CONTENTS

1.0 SCOPE

This document is a User Interface Specification for the Webmaster Modbus/TCP product feature. This is a mapping of the various dynamic variables to their Modbus/TCP register locations.

This document supports the Modbus feature in the following software versions:

S800v004 or earlier S801v003 or earlier

1.1. VERSIONS

Date	Version	Author	Changes
1/8/04	0.3	Ron Gross	Functional Spec
1/19/04	1	Nate Martin	Based on Functional Spec.
2/17/06	2	Charles Johnson	Clarification and Commentary

2.0 INTRODUCTION

The Webmaster product supports TCP/IP communications on 3 different network interfaces (USB, modem, and Ethernet). All configurations of set points are accomplished with a computer running a browser (such as Microsoft Internet Explorer) connected to the WebMaster over one of these interfaces.

The Modbus/TCP option allows the WebMaster to communicate with PC-based applications such as WonderWare and Intellution HMI/SCADA programs, Building Energy Management systems, Distributed Control Systems (DCS), as well as stand-alone HMI devices. The Modbus/TCP information is only available over the Ethernet interface.

The WebMaster is a Modbus Server, meaning that it is only capable of responding to requests from the HMI device. The WebMaster cannot initiate the flow of information, for example, it will not immediately send a new alarm message. It will wait until the HMI device requests the current data contained in specific register locations.

In addition, the HMI software cannot be used to change set points. This is only possible using browser software.

If the HMI device does not directly support Modbus/TCP protocol, then a protocol translation gateway may be required to convert from Modbus/TCP to a protocol that the device supports. Note that Modbus/RTU requires a serial interface, not Ethernet, and therefore is not directly compatible with the WebMaster.

3.0 OVERVIEW

Modbus/TCP is a form of Modbus that uses the TCP/IP layers as a base layer for controlling the communications between different devices.

The Modbus/TCP protocol supports multiple types of data transactions, from reading single bits per transaction, to advanced object-oriented operations. However, to ensure the most compatible system available, the simplest function set is to be made available.

The Modbus/TCP has each transaction type classified in to conformance classes, to ensure consistency and interoperability. Class 0 is the simplest, and allows for reading and writing of multiple 16-bit registers. The Modbus/TCP feature of the WebMaster will support reading of these 16-bit registers, which allows the WebMaster to establish a block of data which contains all the process variables, set points, alarms and input/output statuses that are to be made public to a Modbus/TCP client. This block of data is packaged so that it can be read in 16-bit chunks (or registers) at a time, regardless of the type of data within it. In the following sections, the formatting, storing, and reading of this data are described.

4.0 MODBUS/TCP DRIVER

4.1 MODBUS PROTOCOL

The Modbus protocol, as well as the TCP extension, is well documented in the specifications which are available at http://www.modbus.org, a website established by the Modbus Organization for supporting and organizing the Modbus protocol. Only the use of the protocol is documented here.

4.1.1. TCP

The Modbus/TCP extension includes 7 additional bytes to the original Modbus protocol, which allows for transport over the TCP/IP layers.

MBAP Header	Function Code	Data

The MBAP Header (Modbus Application Protocol Header) consists of 7 bytes of information:

Transaction Identifier	2 bytes	identification of Request/Response transaction – copied from request to response
Protocol Identifier	2 bytes	0 = Modbus protocol
Length	2 bytes	number of following bytes – includes the unit identifier
Unit Identifier	1 byte	identification of remote slave, can be used for broadcasting (not supported)

The Unit Identifier has a special consideration in the WebMaster implementation. If the value is 0, then the request is considered to be a broadcast message; therefore the packet will be processed, and no response will be generated. If the value is anything else, the packet will be processed and a response will be generated.

The broadcast Unit Identifier address is not supported as of this release, as the only function code supported is Read Holding Registers; therefore, a response is required at all times.

4.1.2 Function Codes

The Modbus/TCP feature only supports Function Code 3 (FC3), Read Multiple Registers, which allows the reading of up to 125 16-bit registers, or quantities, within a single request/response cycle. The 125-register limitation is established for the Modbus/TCP standard to maintain consistency with the original Modbus protocol standard, even though a TCP/IP packet can support more data.

Request

Function Code	1 byte	0x03
Starting Address	2 bytes	0x0000 to 0xFFFF
Quantity of Registers	2 bytes	1 to 125 (0x01 to 0x7D)

Response

Function Code	1 byte	0x03
Byte Count	1 byte	2 x N*
Register Values	N* x 2 bytes	

*N = quantity of registers

Error

Function Code	1 byte	0x83
Exception Code	1 byte	

Any other Function Code request will be returned with an error response indicating the Function Code is not supported, as well as a request for too much data or data at a register address that is not present.

4.2 TCP/IP INTERFACE

The Modbus/TCP interface is attached to the TCP/IP stack that is implemented within the WebMaster product, and will listen to all communications that come in on Modbus/TCP registered port 502.

The Modbus/TCP client uses the standard TCP methods for communicating with the driver, as established by the BSD socket interface: connect(), send(), receive() and close(). Up to 10 connections/sockets are possible at one time. If there are 10 active connections, any attempt at any more connections is ignored.

Once a connection has been established, it will be closed after 1 minute of inactivity.

4.3 DATA MODEL

Modbus bases its data model on a series of tables that have individual characteristics. The four primary tables are:

Primary table	data type	type of access
Discrete Inputs	single bit	read-only
Coils	single bit	read-write
Input Registers	16-bits	read-only
Holding Registers	16-bits	read-write

There is no requirement for how the tables are implemented within the product, but the tables are distinctive because of the method that is used to access them within the protocol.

Since only FC3 is supported in the WebMaster implementation, only the Holding Registers-type table is required. To access each entry in to the Holding Register table, a starting address (0 indicates the first entry in the table) is required as well as the number of registers that are requested.

The data storage does not need to be consecutive; in fact, this implementation uses multiple ‘blocks’ within the Holding Register table to support future enhancements and additions to the data without changing the location of the data already present.

The Holding Register table is a large structure that contains smaller structures, each containing the specific types of data, and associated with a defined starting address offset for each type of data. The offset allows the driver to determine if the request needs to access data from the specific structure.

The data within the WebMaster system that is to be made public (process variables, set points, alarms and input/output status) is divided in to four structures (header, alarms, status data, and dynamic data). Any time new data needs to be added to the table (upgraded product, different version, etc), the new data is added to the end of the particular structure that corresponds with that type of data.

The data is stored within the tables local to the Modbus/TCP driver, which allows the driver to quickly access it during a request. The tables are indexed in to during a request using the starting address of the request and the defined offset for that structure.

For example, if the structure containing the headers has an defined offset of 0, and is 128 bytes long and a request for starting address 10 with a length of 5 words is received, the 20^{th} through 29^{th} byte within the table are sent to the client.

Once the data is stored within the local tables, the driver does not differentiate what is stored in them. **The Modbus/TCP client needs to know what data is stored in which register locations to be able to retrieve it, process it, and/or display it.**

4.4 DATA REFRESH

To ensure that the Modbus/TCP client has the most recent data available to it, the Modbus/TCP periodically refreshes the data by reading the selected data and storing it in the specific locations within the tables.

The refresh is performed at a periodic rate to ensure data is new, yet not too often to affect the performance of the WebMaster product.

4.5 DATA ENCODING

Modbus uses a 'big-endian' representation for addresses and data items. This means that when a numerical quantity larger than a single byte is transmitted, the MOST significant byte is sent first. The following sub-topics describe the different types of encoding and show how the data is encoded as it is within the Modbus/TCP packet. Most client drivers will extract the data from the packet in the correct format for use/display within the client environment.

4.5.1 Binary

A binary item is represented as a single bit within a data word. All binary data is packed in to 16-bit data words, which are accessed using FC3 therefore, a single register contains 16 bits of binary data, each having a specific meaning.

value	1^{st}	2^{nd}
0xAA55 (101010100101)	0xAA (10101010)	0x55 (01010101)

4.5.2 16-Bit Word (short)

A 16-bit word item is transmitted with the MOST significant byte first. FC3 reads 16-bit items at a time; therefore, each of these data items will fit within one register that is read.

value	1st	2nd
0x1234	0x12	0x34

4.5.3 32-Bit Word (int)

A 32-bit word item is transmitted with the MOST significant byte first, then the next MOST significant, until all bytes are transmitted. FC3 reads 16-bit items at a time; therefore, two registers are required to read each 32-bit data item.

	1st register		2nd register	
Value	1st	2nd	1st	2nd
0x12345678	0x12	0x34	0x56	0x78

4.5.4 Float Inverse

A float is 32-bits within the WebMaster product; therefore is transmitted just as a 32-bit word item is. FC3 reads 16-bit items at a time; therefore, two registers are required to read each float data item.

	1st register		2nd register	
Value	1st	2nd	1st	2nd
0x12345678	0x12	0x34	0x56	0x78
(as stored in memory)				

4.5.5 Strings

A string is a group of 8-bit data items having a fixed length. The first character of a string is transmitted first, followed by the remaining characters. FC3 reads 16-bit items at a time; therefore, a single register contains two characters of the string. To simply string storage/transfer, each string should be of an even-byte length.

	1st register		2nd register		3rd register		4th register	
value	1st	2nd	1st	2nd	1st	2nd	1st	2nd
'Walchem'	'W'	'a'	'l'	'c'	'h'	'e'	'm'	?

4.6 DATA DICTIONARY

The following tables detail the Modbus addresses required to access each item of the public data.

4.6.1 Addressing (0- or 1-Based)

The addressing within the Modbus/TCP protocol (that is, the data within the physical packet) is 0-based, meaning the first element/item to be accessed is referenced by address 0. The Modbus standard for handling and displaying the data is 1-based, meaning the first element/data item to be access is referenced by address 1.

Most client applications handle this by having the user enter the 1-based number, and then subtract 1 to revert to the 0-based addressing required at the protocol level.

Some client applications allow the user to enter the 0-based number, or a combination, depending on how it is configured.

The addresses defined within the following table are 1-based, as the majority of the client applications work with this method.

4.6.2 Header Data

Header data consists of strings that are available to describe miscellaneous parts of the product. Refer to section **4.5.5 Strings** for the method to extract the string data.

For example, to read the Date item, a Read Holding Register request is generated with address 40033 and a register quantity of 12.

Example of System Summary Information Table:

Header Data Item	Address	Register Quantity	Item Size (bytes)
Controller Name	0001	16	32
Controller Location	0017	16	32
Date	0033	12	24
Software Version Number	0045	16	32
Model Number	0061	16	32
Serial Number	0077	16	32
Controller Phone Number	0093	16	32

4.6.3 Alarm Data

Alarm states are bit-based, with up to 16 alarms encoded within each register. To access an individual alarm state, the register is read (using the address value before the ':'), and the specific bit of the register is checked, with the following rules. Refer to section **4.5.2 16-Bit Word (short)** for the method to properly extract the data.

1001:1:1 defines a single bit, with ':1' indicating the least significant bit of the word and ':16' defining the most significant bit of the word

1005:9-4 defines multiple bits, with :9-4 defining bits 4 through 9 of the word

NOTE:'* unused bits *' place holders are provided to allow for future expansion

For example, to check the Modem Failure Alarm, a Read Holding Register is generated with address 41001 and a register quantity of 1. When the data is returned, and is extracted, it is bit-or'ed with 2 to determine the state.

While Sensor 1-4 is tied to the physical location of the Sensor 1-4 input cards, the Analog and Digital inputs are tied to the software's designation of an input. For example, the first analog input programmed as a Level type of input is Level Analog Input [1].

Alarm Data Item	Address	Bit Count
Modem Failure Alarm	1001:2	1
Ethernet Failure Alarm	1001:3	1
Analog Input Board Failure Alarm	1001:4	1
Digital Input Board Failure Alarm	1001:5	1
Non-Responding Slave Alarm	1001:6	1
* unused bits *	1001:7	1
Pump Failure Alarm	1001:16-8	9
Sensor [1] Board Failure Alarm	1002:1	1
Sensor [1] Sensor Error Alarm	1002:2	1
Sensor [1] Low Alarm	1002:3	1
Sensor [1] High Alarm	1002:4	1
Sensor [1] Calibration Time	1002:5	1
Sensor [1] Temperature Error	1002:6	1
Sensor [1] Temperature Low Alarm	1002:7	1
Sensor [1] Temperature High Alarm	1002:8	1
Sensor [2] Board Failure Alarm	1002:9	1
Sensor [2] Sensor Error Alarm	1002:10	1
Sensor [2] Low Alarm	1002:11	1
Sensor [2] High Alarm	1002:12	1
Sensor [2] Calibration Time	1002:13	1

Sensor [2] Temperature Error	1002:14	1
Sensor [2] Temperature Low Alarm	1002:15	1
Sensor [2] Temperature High Alarm	1002:16	1
Sensor [3] Board Failure Alarm	1003:1	1
Sensor [3] Sensor Error Alarm	1003:2	1
Sensor [3] Low Alarm	1003:3	1
Sensor [3] High Alarm	1003:4	1
Sensor [3] Calibration Time	1003:5	1
Sensor [3] Temperature Error	1003:6	1
Sensor [3] Temperature Low Alarm	1003:7	1
Sensor [3] Temperature High Alarm	1003:8	1
Sensor [4] Board Failure Alarm	1003:9	1
Sensor [4] Sensor Error Alarm	1003:10	1
Sensor [4] Low Alarm	1003:11	1
Sensor [4] High Alarm	1003:12	1
Sensor [4] Calibration Time	1003:13	1
Sensor [4] Temperature Error	1003:14	1
Sensor [4] Temperature Low Alarm	1003:15	1
Sensor [4] Temperature High Alarm	1003:16	1
Level Analog Input [1] Low Alarm	1004:1	1
Level Analog Input [1] Sensor Error	1004:2	1
Level Analog Input [2] Low Alarm	1004:9	1
Level Analog Input [2] Sensor Error	1004:10	1
Level Analog Input [3] Low Alarm	1005:1	1
Level Analog Input [3] Sensor Error	1005:2	1
Level Analog Input [4] Low Alarm	1005:9	1
Level Analog Input [4] Sensor Error	1005:10	1
Level Analog Input [5] Low Alarm	1006:1	1
Level Analog Input [5] Sensor Error	1006:2	1
Level Analog Input [6] Low Alarm	1006:9	1
Level Analog Input [6] Sensor Error	1006:10	1
Level Analog Input [7] Low Alarm	1007:1	1
Level Analog Input [7] Sensor Error	1007:2	1
Level Analog Input [8] Low Alarm	1007:9	1
Level Analog Input [8] Sensor Error	1007:10	1
Generic Analog Input [1] Low Alarm	1008:1	1
Generic Analog Input [1] High Alarm	1008:2	1
Generic Analog Input [1] Sensor Error	1008:3	1
Generic Analog Input [2] Low Alarm	1008:9	1
Generic Analog Input [2] High Alarm	1008:10	1
Generic Analog Input [2] Sensor Error	1008:11	1
Generic Analog Input [3] Low Alarm	1009:1	1
Generic Analog Input [3] High Alarm	1009:2	1
Generic Analog Input [3] Sensor Error	1009:3	1

Generic Analog Input [4] Low Alarm	1009:9	1
Generic Analog Input [4] High Alarm	1009:10	1
Generic Analog Input [4] Sensor Error	1009:11	1
Generic Analog Input [5] Low Alarm	1010:1	1
Generic Analog Input [5] High Alarm	1010:2	1
Generic Analog Input [5] Sensor Error	1010:3	1
Generic Analog Input [6] Low Alarm	1010:9	1
Generic Analog Input [6] High Alarm	1010:10	1
Generic Analog Input [6] Sensor Error	1010:11	1
Generic Analog Input [7] Low Alarm	1011:1	1
Generic Analog Input [7] High Alarm	1011:2	1
Generic Analog Input [7] Sensor Error	1011:3	1
Generic Analog Input [8] Low Alarm	1011:9	1
Generic Analog Input [8] High Alarm	1011:10	1
Generic Analog Input [8] Sensor Error	1011:11	1
Relay Output Timeout Alarm	1012:1-8	8

Flow Meter Analog Input [1] Sensor Error	1012:9	1
Flow Meter Analog Input [1] High Alarm	1012:10	1
Flow Meter Analog Input [1] Low Alarm	1012:11	1
Flow Meter Analog Input [1] Total Alarm	1012:12	1
Flow Meter Analog Input [2] Sensor Error	1013:1	1
Flow Meter Analog Input [2] High Alarm	1013:2	1
Flow Meter Analog Input [2] Low Alarm	1013:3	1
Flow Meter Analog Input [2] Total Alarm	1013:4	1
Flow Meter Analog Input [3] Sensor Error	1013:9	1
Flow Meter Analog Input [3] High Alarm	1013:10	1
Flow Meter Analog Input [3] Low Alarm	1013:11	1
Flow Meter Analog Input [3] Total Alarm	1013:12	1
Flow Meter Analog Input [4] Sensor Error	1014:1	1
Flow Meter Analog Input [4] High Alarm	1014:2	1
Flow Meter Analog Input [4] Low Alarm	1014:3	1
Flow Meter Analog Input [4] Total Alarm	1014:4	1
Flow Meter Analog Input [5] Sensor Error	1014:9	1
Flow Meter Analog Input [5] High Alarm	1014:10	1
Flow Meter Analog Input [5] Low Alarm	1014:11	1

Flow Meter Analog Input [5] Total Alarm	1014:12	1
Flow Meter Analog Input [6] Sensor Error	1015:1	1
Flow Meter Analog Input [6] High Alarm	1015:2	1
Flow Meter Analog Input [6] Low Alarm	1015:3	1
Flow Meter Analog Input [6] Total Alarm	1015:4	1
Flow Meter Analog Input [7] Sensor Error	1015:9	1
Flow Meter Analog Input [7] High Alarm	1015:10	1
Flow Meter Analog Input [7] Low Alarm	1015:11	1
Flow Meter Analog Input [7] Total Alarm	1015:12	1
Flow Meter Analog Input [8] Sensor Error	1016:1	1
Flow Meter Analog Input [8] High Alarm	1016:2	1
Flow Meter Analog Input [8] Low Alarm	1016:3	1
Flow Meter Analog Input [8] Total Alarm	1016:4	1
Level Switch Low Alarm	1018:1-9	7
Generic Counter Rate High Alarm	1020:1-9	9
Generic Counter Rate Low Alarm	1022:1-9	9
Generic Counter Total Alarm	1024:1-9	9
Generic Input Alarm	1026:1-9	9
Flow Meter Digital Input [1] High Alarm	1028:1	1
Flow Meter Digital Input [1] Low Alarm	1028:2	1
Flow Meter Digital Input [1] Total Alarm	1028:3	1
Flow Meter Digital Input [2] High Alarm	1028:9	1
Flow Meter Digital Input [2] Low Alarm	1028:10	1
Flow Meter Digital Input [2] Total Alarm	1028:11	1
Flow Meter Digital Input [3] High Alarm	1029:1	1
Flow Meter Digital Input [3] Low Alarm	1029:2	1
Flow Meter Digital Input [3] Total Alarm	1029:3	1
Flow Meter Digital Input [4] High Alarm	1029:9	1
Flow Meter Digital Input [4] Low Alarm	1029:10	1
Flow Meter Digital Input [4] Total Alarm	1029:11	1
Flow Meter Digital Input [5] High Alarm	1030:1	1
Flow Meter Digital Input [5] Low Alarm	1030:2	1
Flow Meter Digital Input [5] Total Alarm	1030:3	1
Flow Meter Digital Input [6] High Alarm	1030:9	1
Flow Meter Digital Input [6] Low Alarm	1030:10	1
Flow Meter Digital Input [6] Total Alarm	1030:11	1
Flow Meter Digital Input [7] High Alarm	1031:1	1

Flow Meter Digital Input [7] Low Alarm	1031:2	1
Flow Meter Digital Input [7] Total Alarm	1031:3	1
Flow Meter Digital Input [8] High Alarm	1031:9	1
Flow Meter Digital Input [8] Low Alarm	1031:10	1
Flow Meter Digital Input [8] Total Alarm	1031:11	1
Flow Meter Digital Input [9] High Alarm	1032:1	1
Flow Meter Digital Input [9] Low Alarm	1032:2	1
Flow Meter Digital Input [9] Total Alarm	1032:3	1

4.6.4 Status Data

Status data generally consists of 16-bit words, 32-bit longs or floats. To access an individual Status Data item, 1 or 2 registers are required to be read. Refer to sections **4.5.2 16-Bit Word (short), 4.5.3 32-Bit Word (int)** and **4.5.4 Float Inverse** for the methods to properly extract the data. The following rules indicate the format of the table:

Address — defines the starting address to read to access the first element (or only element) of the item

Register Count (Item) — defines the number of registers to read to access a single element of the item

Register Count (Total) — defines the number of registers to read to access all elements of the item

Item Count — defines the number of elements within the full item

Item [6-1] — defines the item is an array of elements, with *item* [1] being accessed first (first set of registers).

For example, to check the second element of Flow meter Analog Input [2] Status, a Read Holding Register is generated with address 42051 (42049 + 2) and a register quantity of 2.

While Sensor 1-4 is tied to the physical location of the Sensor 1-4 input cards, the Analog and Digital inputs are tied to the software's designation of an input. For example, the first analog input programmed as a Level type of input is Level Analog Input [1].

Status Data Item	Address	Register Count (Item)	Register Count (Total)	Item Count
Sensor [1-4] Status	2001	2	8	4
Temperature [1-4] Status	2009	2	8	4
Generic Analog Input [1-8] Status	2017	2	16	8
Level Analog Input [1-8] Status	2033	2	16	8
Flow meter Analog Input [1-8] Status	2049	2	16	8
Flow meter Digital Input [1-9] Status	2065	2	18	9

Relay Output [1]Control Mode (Hand-Off-Auto)	2083	2	2	1
Relay Output [1] On-Time	2085	1	1	1
SV.OutputStatus [2].OutMode	2086	2	2	1
SV.OutputStatus [2].On	2088	1	1	1
SV.OutputStatus [3].OutMode	2089	2	2	1
SV.OutputStatus [3].On	2091	1	1	1
SV.OutputStatus [4].OutMode	2092	2	2	1
SV.OutputStatus [4].On	2094	1	1	1
SV.OutputStatus [5].OutMode	2095	2	2	1
SV.OutputStatus [5].On	2097	1	1	1
SV.OutputStatus [6].OutMode	2098	2	2	1
SV.OutputStatus [6].On	2100	1	1	1
SV.OutputStatus [7].OutMode	2101	2	2	1
SV.OutputStatus [7].On	2103	1	1	1
SV.OutputStatus [8].OutMode	2104	2	2	1
SV.OutputStatus [8].On	2106	1	1	1
4-20 mA Output Status [4-1]	2107	2	8	4
Digital Input State [A-C, 1-6]	2115	1	1	1
SV.Connection.DialedManually	2119	1	1	1

The status message is encoded using the following values.

For Sensor inputs, Temperature inputs, Digital inputs and Analog inputs:

0	
1	Normal
2	Off
3	On
4	Ok
5	Self Test
6	Wait
7	Sampling
8	Hold
9	Sensor Error
10	High Alarm
11	Low Alarm
12	Calibration Time
13	Board Failure
14	Pump Failure
15	Total Alarm

For Analog outputs and Relay outputs:

0	
1	Off
2	On
3	Time Out
4	A/D Startup
5	Hand
6	Manual Off
7	Failure
8	Invalid
9	Calibrate
10	Calibrate Sen 1
11	Calibrate Sen 2
12	Calibrate Sen 3
13	Calibrate Sen 4
14	Normal
15	Overrange
16	Underrange
17	Loop Cal
18	Sensor Error
19	Internal Lock
20	Unknown
21	DI A Lockout
22	DI B Lockout
23	DI C Lockout
24	DI 1 Lockout
25	DI 2 Lockout
26	DI 3 Lockout
27	DI 4 Lockout
28	DI 5 Lockout
29	DI 6 Lockout

30	Relay 1 Lockout
31	Relay 2 Lockout
32	Relay 3 Lockout
33	Relay 4 Lockout
34	Relay 5 Lockout
35	Relay 6 Lockout
36	Relay 7 Lockout
37	Relay 8 Lockout
38	No Sensor Selected
39	Waiting
40	Sampling
41	Holding
42	Blowdown
43	No Posi-Flow Configured
44	Units Mismatch
45	Disp. Lockout
46	Bio Lockout
47	PreBleed Lockout
48	Pre-Bleed
49	Waiting
50	On Delay
51	Pending
52	Bleed Lockout Time
53	Bio Add
54	Dispersant Add

Relay Output Modes	Digital Input States
0 = HAND	256 = On
1 = OFF	0 = Off
2 = AUTO	

4.6.5 Dynamic Data

Dynamic data generally consists of 16-bit words, 32-bit longs or floats. To access an individual Dynamic Data item, 1 or 2 registers are required to be read. Refer to sections **4.5.2 16-Bit Word (short), 4.5.3 32-Bit Word (int)** and **4.5.4 Float Inverse** for the methods to properly extract the data. The following rules indicate the format of the table:

Address	defines the starting address to read to access the first element (or only element) of the item
Register Count (Item)	defines the number of registers to read to access a single element of the item
Register Count (Total)	defines the number of registers to read to access all elements of the item
Item Count	defines the number of elements within the full item
Item [6-1]	defines the item is an array of elements, with *item* [1] being accessed first (first set of registers).

For example, to check the item Sensor [1] Current Reading, a Read Holding Register is generated with address 43001 and a register quantity of 2.

While Sensor 1-4 is tied to the physical location of the Sensor 1-4 input cards, the Analog and Digital inputs are tied to the software's designation of an input. For example, the first analog input programmed as a Level type of input is Level Analog Input [1].

Dynamic Data Item	Address	Register Count (Item)	Register Count (Total)	Item Count
Sensor [1] Current Reading	3001	2	2	1
Sensor [1] Uncalibrated Reading	3003	2	2	1
Sensor [1] mV Output	3005	2	2	1
Sensor [1] Temperature Reading	3007	2	2	1
Sensor [1] Uncalibrated Temperature	3009	2	2	1
* unused bits *	3011	2	2	1
Sensor [2] Current Reading	3013	2	2	1
Sensor [2] Uncalibrated Reading	3015	2	2	1
Sensor [2] mV Output	3017	2	2	1
Sensor [2] Temperature Reading	3019	2	2	1
Sensor [2] Uncalibrated Temperature	3021	2	2	1
* unused bits *	3023	2	2	1
Sensor [3] Current Reading	3025	2	2	1
Sensor [3] Uncalibrated Reading	3027	2	2	1
Sensor [3] mV Output	3029	2	2	1
Sensor [3] Temperature Reading	3031	2	2	1
Sensor [3] Uncalibrated Temperature	3033	2	2	1
* unused bits *	3035	2	2	1
Sensor [4] Current Reading	3037	2	2	1

Sensor [4] Uncalibrated Reading	3039	2	2	1
Sensor [4] mV Output	3041	2	2	1
Sensor [4] Temperature Reading	3043	2	2	1
Sensor [4] Uncalibrated Temperature	3045	2	2	1
* unused bits *	3047	2	2	1
4-20 mA Input [8-1] Uncalibrated mA Reading	3049	2	16	8
4-20 mA Input [8-1] Calibrated Present mA	3065	2	16	8
Generic 4-20 [8-1] Present Value	3081	2	16	8
Generic 4-20 [8-1] mA?	3097	2	16	8
Level 4-20 [8-1] Present Volume	3113	2	16	8
* unused bits *	3129	4	4	1
PosiFlow [A] Total	3133	2	2	1
PosiFlow [A] Date/Time of Last Total Reset	3135	16	16	1
* unused bits *	3129	4	4	1
PosiFlow [B] Total	3155	2	2	1
PosiFlow [B] Date/Time of Last Total Reset	3157	16	16	1
* unused bits *	3129	4	4	1
PosiFlow [C] Total	3177	2	2	1
PosiFlow [C] Date/Time of Last Total Reset	3179	16	16	1
* unused bits *	3129	4	4	1
PosiFlow [1] Total	3199	2	2	1
PosiFlow [1] Date/Time of Last Total Reset	3201	16	16	1
* unused bits *	3129	4	4	1
PosiFlow [2] Total	3221	2	2	1
PosiFlow [2] Date/Time of Last Total Reset	3223	16	16	1
* unused bits *	3129	4	4	1
PosiFlow [3] Total	3243	2	2	1
PosiFlow [3] Date/Time of Last Total Reset	3245	16	16	1
* unused bits *	3129	4	4	1
PosiFlow [4] Total	3265	2	2	1
PosiFlow [4] Date/Time of Last Total Reset	3267	16	16	1
* unused bits *	3129	4	4	1
PosiFlow [5] Total	3287	2	2	1
PosiFlow [5] Date/Time of Last Total Reset	3289	16	16	1
* unused bits *	3129	4	4	1
PosiFlow [6] Total	3309	2	2	1
PosiFlow [6] Date/Time of Last Total Reset	3311	16	16	1
4-20 mA Output [1] Scaled Input Value	3327	2	2	1
4-20 mA Output [1] mA Output	3329	2	2	1
4-20 mA Output [2] Scaled Input Value	3331	2	2	1
4-20 mA Output [2] mA Output	3333	2	2	1
4-20 mA Output [3] Scaled Input Value	3335	2	2	1
4-20 mA Output [3] mA Output	3337	2	2	1
4-20 mA Output [4] Scaled Input Value	3339	2	2	1
4-20 mA Output [4] mA Output	3341	2	2	1
Min, Max and Ave Calculated Over x Hours	3344	1	1	1

Sensor [1] Average Reading	3345	2	2	1
Sensor [1] Minimum Reading	3347	2	2	1
Sensor [1] Maximum Reading	3349	2	2	1
Sensor [2] Average Reading	3351	2	2	1
Sensor [2] Minimum Reading	3353	2	2	1
Sensor [2] Maximum Reading	3355	2	2	1
Sensor [3] Average Reading	3357	2	2	1
Sensor [3] Minimum Reading	3359	2	2	1
Sensor [3] Maximum Reading	3361	2	2	1
Sensor [4] Average Reading	3363	2	2	1
Sensor [4] Minimum Reading	3365	2	2	1
Sensor [4] Maximum Reading	3367	2	2	1
Generic AI [1] Average Reading	3369	2	2	1
Generic AI [1] Minimum Reading	3371	2	2	1
Generic AI [1] Maximum Reading	3373	2	2	1
Generic AI [2] Average Reading	3375	2	2	1
Generic AI [2] Minimum Reading	3377	2	2	1
Generic AI [2] Maximum Reading	3379	2	2	1
Generic AI [3] Average Reading	3381	2	2	1
Generic AI [3] Minimum Reading	3383	2	2	1
Generic AI [3] Maximum Reading	3385	2	2	1
Generic AI [4] Average Reading	3387	2	2	1
Generic AI [4] Minimum Reading	3389	2	2	1
Generic AI [4] Maximum Reading	3391	2	2	1
Generic AI [5] Average Reading	3393	2	2	1
Generic AI [5] Minimum Reading	3395	2	2	1
Generic AI [5] Maximum Reading	3397	2	2	1
Generic AI [6] Average Reading	3399	2	2	1
Generic AI [6] Minimum Reading	3401	2	2	1
Generic AI [6] Maximum Reading	3403	2	2	1
Generic AI [7] Average Reading	3405	2	2	1
Generic AI [7] Minimum Reading	3407	2	2	1
Generic AI [7] Maximum Reading	3409	2	2	1
Generic AI [8] Average Reading	3411	2	2	1
Generic AI [8] Minimum Reading	3413	2	2	1
Generic AI [8] Maximum Reading	3415	2	2	1
Level AI [1] Average Reading	3417	2	2	1
Level AI [1] Minimum Reading	3419	2	2	1
Level AI [1] Maximum Reading	3421	2	2	1
Level AI [2] Average Reading	3423	2	2	1
Level AI [2] Minimum Reading	3425	2	2	1
Level AI [2] Maximum Reading	3427	2	2	1
Level AI [3] Average Reading	3429	2	2	1
Level AI [3] Minimum Reading	3431	2	2	1
Level AI [3] Maximum Reading	3433	2	2	1
Level AI [4] Average Reading	3435	2	2	1
Level AI [4] Minimum Reading	3437	2	2	1

Level AI [4] Maximum Reading	3439	2	2	1
Level AI [5] Average Reading	3441	2	2	1
Level AI [5] Minimum Reading	3443	2	2	1
Level AI [5] Maximum Reading	3445	2	2	1
Level AI [6] Average Reading	3447	2	2	1
Level AI [6] Minimum Reading	3449	2	2	1
Level AI [6] Maximum Reading	3451	2	2	1
Level AI [7] Average Reading	3453	2	2	1
Level AI [7] Minimum Reading	3455	2	2	1
Level AI [7] Maximum Reading	3457	2	2	1
Level AI [8] Average Reading	3459	2	2	1
Level AI [8] Minimum Reading	3461	2	2	1
Level AI [8] Maximum Reading	3463	2	2	1
Flow meter AI [1] Average Reading	3465	2	2	1
Flow meter AI [1] Minimum Reading	3467	2	2	1
Flow meter AI [1] Maximum Reading	3469	2	2	1
Flow meter AI [2] Average Reading	3471	2	2	1
Flow meter AI [2] Minimum Reading	3473	2	2	1
Flow meter AI [2] Maximum Reading	3475	2	2	1
Flow meter AI [3] Average Reading	3477	2	2	1
Flow meter AI [3] Minimum Reading	3479	2	2	1
Flow meter AI [3] Maximum Reading	3481	2	2	1
Flow meter AI [4] Average Reading	3483	2	2	1
Flow meter AI [4] Minimum Reading	3485	2	2	1
Flow meter AI [4] Maximum Reading	3487	2	2	1
Flow meter AI [5] Average Reading	3489	2	2	1
Flow meter AI [5] Minimum Reading	3491	2	2	1
Flow meter AI [5] Maximum Reading	3493	2	2	1
Flow meter AI [6] Average Reading	3495	2	2	1
Flow meter AI [6] Minimum Reading	3497	2	2	1
Flow meter AI [6] Maximum Reading	3499	2	2	1
Flow meter AI [7] Average Reading	3501	2	2	1
Flow meter AI [7] Minimum Reading	3503	2	2	1
Flow meter AI [7] Maximum Reading	3505	2	2	1
Flow meter AI [8] Average Reading	3507	2	2	1
Flow meter AI [8] Minimum Reading	3509	2	2	1
Flow meter AI [8] Maximum Reading	3511	2	2	1
Flow meter AI [1] Total	3513	2	2	1
Flow meter AI [1] Rate	3515	2	2	1
Flow meter AI [2] Total	3517	2	2	1
Flow meter AI [2] Rate	3519	2	2	1
Flow meter AI [3] Total	3521	2	2	1
Flow meter AI [3] Rate	3523	2	2	1

Flow meter AI [4] Total	3525	2	2	1
Flow meter AI [4] Rate	3527	2	2	1
Flow meter AI [5] Total	3529	2	2	1
Flow meter AI [5] Rate	3531	2	2	1
Flow meter AI [6] Total	3533	2	2	1
Flow meter AI [6] Rate	3535	2	2	1
Flow meter AI [7] Total	3537	2	2	1
Flow meter AI [7] Rate	3539	2	2	1
Flow meter AI [8] Total	3541	2	2	1
Flow meter AI [8] Rate	3543	2	2	1
Flow Meter Digital Input [A] Total	3545	2	2	1
Flow Meter Digital Input [A] Rate	3547	2	2	1
Flow Meter Digital Input [B] Total	3549	2	2	1
Flow Meter Digital Input [B] Rate	3551	2	2	1
Flow Meter Digital Input [C] Total	3553	2	2	1
Flow Meter Digital Input [C] Rate	3555	2	2	1
Flow Meter Digital Input [1] Total	3557	2	2	1
Flow Meter Digital Input [1] Rate	3559	2	2	1
Flow Meter Digital Input [2] Total	3561	2	2	1
Flow Meter Digital Input [2] Rate	3563	2	2	1
Flow Meter Digital Input [3] Total	3565	2	2	1
Flow Meter Digital Input [3] Rate	3567	2	2	1
Flow Meter Digital Input [4] Total	3569	2	2	1
Flow Meter Digital Input [4] Rate	3571	2	2	1
Flow Meter Digital Input [5] Total	3573	2	2	1
Flow Meter Digital Input [5] Rate	3575	2	2	1
Flow Meter Digital Input [6] Total	3577	2	2	1
Flow Meter Digital Input [6] Rate	3579	2	2	1
RSI	3581	2	2	1
LSI	3583	2	2	1