

SIEMENS

Maxum edition II

PD PA AP
Modbus User's Guide

Equipment Manual




<u>Overview</u>	1
<u>Modbus Address Map</u>	2
<u>Host/Analyzer Messages</u>	3
<u>Modbus Protocol Reference</u>	4
<u>Appendix A - Contact Information</u>	A
<u>Appendix B - Change Log</u>	B

Modbus Reference excerpt from GCP Reference Manual. Full Modbus TCP setup information can be found in the GCP Reference Manual, A5E03944542001

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

 DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.
 WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.
 CAUTION
indicates that minor personal injury can result if proper precautions are not taken.
NOTICE
indicates that property damage can result if proper precautions are not taken.


If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

 WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Table of contents

1	Overview	7
1.1	Security information	7
1.2	Definitions	7
1.3	Modbus Operation.....	8
1.4	Hardware Configuration	9
1.5	Serial Communication	9
1.6	Modbus TCP Configuration.....	10
1.7	Unit/Stream/Component Limits	10
1.8	Component Values.....	11
1.9	Scaled Results	11
1.10	Floating Point Results	11
1.11	Status Information	11
1.12	Analyzer Alarms	12
1.13	NAU Alarms	12
2	Modbus Address Map	13
2.1	Address Map Description	13
2.2	General Address Map Rules	13
2.3	Address Map Limits.....	14
2.4	Creating and Loading an Address Map.....	14
2.5	Address Map Entries.....	15
2.6	Sample Address Entries.....	17
2.7	Example Address Map Configuration.....	18
2.8	Viewing and Editing the Address Map	19
2.9	Configure Analyzers to Transmit Results.....	19
2.10	Special Instructions	20
3	Host/Analyzer Messages	21
3.1	Summary of Host/Analyzer Communication	21
3.2	AIREAD	21
3.3	ALARM.....	22
3.4	ANALYZERSTATUS	22
3.5	CALIBRATE	23

3.6	CLEARALARM.....	23
3.7	CURRENTSTREAM.....	23
3.8	CYCLELENGTH.....	24
3.9	DATE and Time.....	24
3.10	DCHG.....	25
3.11	DEDICATEDSTREAM	25
3.12	DIREAD.....	25
3.13	DOREAD.....	25
3.14	DOSET	25
3.15	ECHG.....	25
3.16	EUHI.....	26
3.17	PROGRAMRUN.....	26
3.18	RDME.....	26
3.19	RESULT	26
3.20	SCMIN, SCSEC, SCHR, SCDAY, SCMON, SCYR	27
3.21	SELECTSTREAM	27
3.22	SKIPSTREAM	27
3.23	STANDBY	28
4	Modbus Protocol Reference	29
4.1	Protocol Formats.....	29
4.2	RTU vs. ASCII vs. TCP.....	29
4.3	Modicon Types.....	29
4.4	IEEE 32 Bit Float Format	30
4.5	EUHI 16 Bit Float Format	31
4.6	16-Bit Conversion Routines	32
4.7	Communication Errors	33
4.8	Implementation of Modbus Protocol.....	33
4.9	Modbus Message Format	34
4.10	Message Length.....	34
4.11	Message Content	35
4.12	Digital Values	35
4.13	RTU Checksum.....	36
4.14	Basic Data Types	36
4.15	Checksum and Communication Errors	36
4.16	CRC Lookup Table.....	37

4.17	Character Transmission	37
4.18	Function Codes	38
4.19	Exception Response Codes	39
4.20	Error Responses	39
4.21	Function Reference	40
4.21.1	Function 01 - Read Coil	40
4.21.2	Function 02 - Read Input	41
4.21.3	Function 03 - Read Output	42
4.21.4	Function 04 - Read Input	43
4.21.5	Function 05 - Set Single Coil	43
4.21.6	Function 08 - Loopback Diagnostic	44
4.21.7	Function 15 - Set Multiple Coils	47
4.21.8	Function 16 - Set Multiple Registers	48
A	Appendix A - Contact Information	51
A.1	Product Information Availability	51
A.2	Contacts	52
B	Appendix B - Change Log	53
B.1	December 2019 Changes	53

Overview

1.1 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (<https://www.siemens.com/industrialsecurity>).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (<https://www.siemens.com/industrialsecurity>).

1.2 Definitions

Term	Definition
Analyzer	Any of the following units that support the Modbus slave protocol: Maxum I, Maxum II, NAU, Advance Plus, MicroSAM
Coil	Modicon term for a Boolean value, also described as a flag
Float	A data element representing an Analog value using IEEE standard 32-bit floating point format
DCS	Host computer that is the Master
HCI-H	(Host Computer Communications Interface – HIWAY) The Advance Optichrom system's Modbus protocol interface
Host	An external computer system which acts as the Modbus master and requests data from the Slave analyzer.
Master	Modbus systems require one master device (host) which sends requests to one or more slave devices
Modbus	Communications protocol, defined in 1979 for Modicon Programmable Logic Controllers, that has become a de facto standard for data communications.

Modbus TCP	Modbus variant used for communications over TCP/IP networks (different from Modbus over TCP). Supported for software versions 5.0.7 and higher.
NAU	(Network Access Unit) Provides general purpose I/O for Siemens GC systems. Any analyzer can be configured for this purpose, but it is recommended that it be done on a dedicated unit.
Optichrom	Siemens Advance Optichrom product line of gas analyzers and network devices, the predecessor to the Siemens Maxum product line
Register	term for a 16-bit 2's complement integer value, also described as a Word
RTU	(Remote Terminal Unit) Modbus RTU is a format where values are transmitted in binary form using serial communication links.
Slave	Modbus device that passively waits for requests from a Modbus master device

1.3 Modbus Operation

From GC

The following steps are illustrated in the figure below.

- The analyzer sends sql commands or Optichrom units send Advance Data Hiway messages to the NAU database at the end of a cycle or after a manual transmit.
- The NAU updates the Modbus_addmap table. The correct address is identified and updated with the value received from the analyzer. Identification is based on the value_type, anlz, application, stream, and result.
- The Modbus_Addmap table sends the value to the MODBUS communication software..

From the DCS

- The DCS sends a message to the MODBUS communication software
- The MODBUS communication software processes a read request and returns a message to the DCS or sends the value to the modbus_addmap table.
- The MODBUS_addmap table determines the correct message to send to a specific analyzer based on the address value_type, anlz, application, stream, and result.

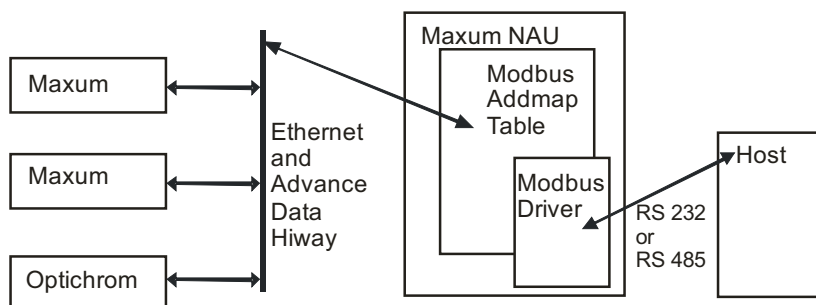


Figure 1-1 Modbus Operation

1.4 Hardware Configuration

To configure hardware, have:

- GCs and a Network Access Unit (NAU) attached to the same ethernet or Advance DataNET network. Small systems, like MicroSAMs, can allow the analyzer to do its own Modbus communications. In addition, Modbus TCP allows an analyzer to do its own Modbus communications.
- Host DCS computer, setup to be a MODBUS master using Modbus RTU messages and/or Modbus TCP messages. Host connects to the NAU/Analyzer using the RS-232 or RS-485 serial ports for Modbus RTU, or using Ethernet ports when using Modbus TCP.

1.5 Serial Communication

To set up serial communication:

These settings can be changed at any time prior to or during MODBUS communication. MicroSAM requires a reset after changing any settings.

For MicroSAM, the unit must be configured for Modbus serial communication at startup. Consult MicroSAM documentation for details.

Using Gas Chromatograph Portal, for all versions:

In the Analyzer View of the Analyzer Portal, choose Network on the Navigation menu and then Serial Settings.

The serial ports will be displayed. These are hard coded depending on the type of hardware installed.

SYSCON1 - Port 1 is dedicated to Modbus

SYSCON2 - Ports 1, 3, and 4 are dedicated to Modbus

CIM - Port 1 is dedicated to Modbus

On the far right side of the display pane, set the details as needed for each serial port to be used for serial Modbus communications. The drop-down menus for each attribute show the possible values.

Using System Manager:

For versions prior to 5.0.7:

Using System Manager, enter a value in the modbus_setting attribute of the System_Control table:

1.7 Unit/Stream/Component Limits

Y: Baud, Parity, Data bits, Stop bits, Flow control, RS

Baud: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200

Parity: n, o, or e (none, odd, or even)

Data bits: 7 or 8

Stop bits: 1 or 2

Flow control: n or h (none or hardware)

RS: 232 or 485

Default: 1:19200,n,8,1,n,232

For versions 5.0.7 and above:

Use the Serial_Settings table to configure 4 port or 2 port units.

a,b,c,d,e,f (example--1:19200,n,8,1,n,232)

where

a = baud rate (50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600 or 115200)

b = parity (e for even, o or odd, or n for none)

c = data bits (7 or 8)

d = stop bits (1 or 2)

e = flow control (h for hardware, n for none)

f = 232 for RS-232, 485 for RS-485

1.6 Modbus TCP Configuration

Modbus TCP Configuration

Modbus TCP communication is enabled by placing "TCP" in the modbus_settings attribute of the System_Control table. This can be done by editing the System_Control table in Maxum System Manager. It can also be accomplished by checking the "Enable Modbus TCP" checkbox in GCP software (to access this checkbox, under the Analyzer tab choose the Network selection on the Navigation Menu).

When using Modbus TCP, the master(s) must be configured to use port 502 and the IP address of the unit containing the Modbus tables. The slave address is not used for Modbus TCP. Multiple masters can access the Modbus data on a single GC/NAU. The Maximum number of Modbus TCP masters (connections) is 16.

1.7 Unit/Stream/Component Limits

(Software Version 4.0 and above)

There are no limits on the number of GCs/units. Streams must have stream numbers ≤ 100 . Results must have result # < 500 . There are no requirements for GC numbering.

Component numbers should be consecutive integers. Component numbers are assigned in the order transmitted for Optichrom Advance or correspond to the `trtval` setting in the result table for GCs.

1.8 Component Values

Result values (any component or other value stored in the Analyzer's result table) can be sent to the Host as either:

- scaled integers, or
- 32-bit floating point numbers.

1.9 Scaled Results

Scaled results are stored in a single register as:

$$\text{scaled result} = \text{Round}(\text{ScaleFactor} * \text{ResultValue} / \text{EUHI})$$

Note: EUHI must be ≥ 1.0

1.10 Floating Point Results

A floating point result is stored in a pair of registers. Only the first register in the pair is defined in the address map. The floating point format for Modbus addresses in the 70000s is the 754 standard IEEE 32-bit float. The format for addresses in the 20000s is IEEE 32-bit float with the words swapped. The format for addresses in the 40000s is either swapped or unswapped based on the `data_type` setting (R or Q). Addresses for floating point results must be numbered as odd numbers.

Floating point values can be transmitted from Advance Optichrom by using a fractional full scale of 1.0 and an EUHI of 1.0.

1.11 Status Information

Each analyzer transmits status information at the end of the application cycle, along with the results. This information is made available to the Host to ascertain the following:

- Whether the application is in normal state, reporting data on schedule
- Whether the application is calibrating, in hold, out of service, or has not reported data on schedule
- Whether the application has an error in the current cycle
- Whether the application is operating with a dedicated stream

1.13 NAU Alarms

- The enable/disable state of every stream in an application
- The current and next stream to be analyzed
- Whether specific database or EUHI values have changed

1.12 Analyzer Alarms

Analyzer alarms are defined as any alarms that may occur on the GCs. Most alarms that are associated with Modbus communications will appear on the Network Access Unit (NAU).

Alarm	Meaning
-------	---------

698	The NAU is unreachable or refuses connection. Results will not be transmitted.
-----	--

1.13 NAU Alarms

NAU alarms are defined as any alarms that may occur on the Slave unit.

Alarm	Meaning
-------	---------

699	The analyzer tried to transmit a result or analyzer status that was not defined in the address map.
-----	---

700	The analyzer is not reachable or refuses connection. It will not receive a command from the Host.
-----	---

701	Scale factor or EUHI is missing.
-----	----------------------------------

703	The Host attempted to set an address that was not defined in the address map.
-----	---

704	The Host has attempted a write to an address that is read-only.
-----	---

705	Host command could not be directed to an address in the address map.
-----	--

706	Can't locate EUHI.
-----	--------------------

709	The Data_type does not match the value_type. Correction has been taken.
-----	---

Modbus Address Map

2.1 Address Map Description

To have working communication among the GC(s), Modbus, and Host, you must have an address map customized to user specifications. The Modbus Address Map is a text file used for loading the Maxum Network Access Unit (NAU) with details for processing the following:

- GC results to be placed at certain addresses so the host computer can read them.
- Control requests sent from the DCS to be directed to analyzers.

The map contains details about all addresses configured to send and receive data. Since no assumptions are made about what is contained at an certain address, the data type and a value type indicate what kind of information is stored at an address and any actions to be taken. For example, a RDME flag is set to "True" from the analyzer and reset to "False" automatically. The value column in the Modbus_addmap table in the Gas Chromatography Portal contains information in readable format, not the format the Modbus Driver sends to the Host. A scaled result will be in the pre-scaled format, as it appears in the result table.

An address map may be configured by Siemens. according to customer requirements or configured by the customer – either by using a general default map or by editing that map to meet specific requirements. Microsoft Excel is used as the configuration tool for Version 3.0, by editing the map and saving it in comma delimited (.CSV) format. The resulting text file can be loaded onto the NAU by using System Manager/Tools/Utilities/Loader/MODBUS Load, or DBConverter can be used in software version 4.3 and higher. An unload utility is also available.

2.2 General Address Map Rules

Each transmitted result must be defined in the address map, and each transmitting application must have a status in the map.

For every result that the system needs to handle, have the following available:

- The analyzer number
- Application number
- Stream number
- Result number (trtval value)

In addition, have on hand the specific customer requirements for host computer controls and information.

(Software Version 4.0 and Above) As the map is loaded onto the analyzer using Maxum Utilities/Loader/Modbus Load, duplicate addresses that contain the same information will be removed. That is, two addresses for result 1/ analyzer 14/application 1/stream 2 are not allowed and will be removed during loading of the map.

2.3 Address Map Limits

The address map is limited to 4000-4500 addresses. This will be sufficient to handle an old-style Optichrom HCI-H map for 18 applications/ 9 streams/ 9 results. A custom address map may have any combination of types of values, with the only requirement being a result or result plus EUHI for every component transmitted to the NAU and an Analyzer status for every application tagged to transmit results. A much larger map may be successfully loaded, but, when the NAU goes through a reset, the database cannot be rebuilt.

2.4 Creating and Loading an Address Map

Use a text editor or Microsoft Excel to create a map.

Load the map onto the NAU.

Use the following to configure an analyzer to transmit results:

1. Mark results for transmit using the trtval and EUHI attributes of the result table.
2. Mark stream for transmission using the autotr attribute in the stream_method table.
3. Designate NAU to receive results, using the host table

type:6, anlzref:NAU's anlz_id

Or

Version 3.1 and Above: The following format was developed for use by NAUs that are used to gather results from 3rd party analyzers and transmit them to a host. This will allow for running a MaxBasic program that transmits by frequency using the trtnow attribute in the stream_method table. An application and streams will need to be created to create the results in the result table. Store AI or other values in result table. Set up additional results with these specific result names:

Analyzerstatus – set to status as in HCI-H (required)

Currentstream – set saved_value to the next stream

Standby - set to 1 for running, 0 for hold

Alarmstream - set for stream alarm

Alarmanlz – set to anlz alarm

Alarmapp – set of application alarm

Dedicatedstream – set to dedicated stream

Cyclelength – set to cycle length

Skipstream – set saved value to stream that is enabled(positive)
or disabled(negative)

Calibrate – set to 1 if in calibration

Do NOT mark these for transmit, only mark the result's trtvals. Set values in these results.

Use type 7 for the host. This is a new free format used for Modbus, which allows you to be free of entries in the actual tables. Example:

1&&2	1	analyzerstatus	NULL	NULL	NULL	NULL	NULL	1000.000000
1&&2	2	currentstream	NULL	NULL	NULL	NULL	NULL	4.000000
1&&2	3	standby	NULL	NULL	NULL	NULL	NULL	0.000000
1&&2	4	alarmstream	NULL	NULL	NULL	NULL	NULL	362.000000
1&&2	5	result1	NULL	1	NULL	NULL	NULL	0.234000
1&&2	6	result2	NULL	2	NULL	NULL	NULL	5.678000
1&&2	7	alarmapp	NULL	NULL	NULL	NULL	NULL	562.000000
1&&2	8	alamranlz	NULL	NULL	NULL	NULL	NULL	444.000000
1&&2	9	enablestream	NULL	NULL	NULL	NULL	NULL	1.000000
1&&2	10	skipstream	NULL	NULL	NULL	NULL	NULL	1.000000
1&&2	11	skipstream	NULL	NULL	NULL	NULL	NULL	2.000000
1&&2	12	skipstream	NULL	NULL	NULL	NULL	NULL	-3.000000
1&&2	13	skipstream	NULL	NULL	NULL	NULL	NULL	-4.000000
1&&2	14	skipstream	NULL	NULL	NULL	NULL	NULL	-5.000000
1&&2	15	skipstream	NULL	NULL	NULL	NULL	NULL	-6.000000
1&&2	16	skipstream	NULL	NULL	NULL	NULL	NULL	-7.000000

Figure 2-1 Address Map Spreadsheet

This will transmit 2 results, along with supporting information to the host.

2.5 Address Map Entries

The entries in the address map are detailed in the following table:

Field Name	Type	Range/Description
Ref_Index	Integer	Arbitrary, suggest using sequential integers, from 1 up, as line numbers
Host_Tag Name	Character	Arbitrary, for use in referencing host database. This is the label assigned by the host to the data item
Description	Character	If blank, will be set by the load routine. As results transmit, this will contain the name from the result table.
result_name	Character	(not used)
Data_Type	Character	S for scaled fraction (only positive values) E for 16-bit EUHI float B for boolean R for real number (IEEE 32-bit float) Q same R words swapped I for an integer that can be positive or negative D floating point 64 bit date

2.5 Address Map Entries

Modbus_address	Integer	For digital: 00001 to 9999 10001 to 19999 For integer or float: 20001 TO 29999 30001 to 39999 40001 to 49999 70001 to 79999 A pair of addresses (numbered as odd numbers) are required for each 32-bit floating point number. All addresses must be unique within each of the four tables (0, 1, 3, and 4). The first digit of the address specifies the table. Addresses in the 20000s, 40000s and 70000s share the same memory locations. These address ranges allow the host to format the data into swapped 32-bit, 16-bit integer, or 32-bit float, respectively.
Anlz	Integer	Analyzer id/unit
Application	Integer	Must be a valid application id for the given analyzer.
Stream	Integer	Must be a valid stream id for the given application.
Result	Integer	Must be a valid result.

Value_type	Character	(in many cases, only the first 2 characters are required, see bold – Values are not case sensitive) HOSTALIVE = host alive flag RESULT = analyzer result value CYCLELENGTH = length of cycle EUHI = Euhi RDME = readme flag SELECTSTREAM = stream force flag SKIPSTREAM = stream skip flag DEDICATEDSTREAM = dedicated stream CURRENTSTREAM = current stream flag ANALYZERSTATUS = analyzer status DCHG = database change flag ECHG = euhi change flag STANDBY = standby flag CALIBRATE = calibration flag PROGRAMRUN = execute event DOSET = set a DO on the NAU DIREAD = read a DI on the NAU DOREAD = read a DO on the NAU ALARM = fault alarm code CLEARALARM = clear Maxum alams Clock settings: SCMIN, SCHR, SCDAY, SCMON, SCYR DATETIME – replaces DATE and TIME in Version 5.0.
Initvalue	Character	Initial value (not for results or EUHI)
Slave_address	Integer	Modbus Slave address
Euhi_address	Integer	Modbus address for EUHI

There are two special entries in the address map that set the "bad value" and the scale factor. The scale factor will be used to calculate all scaled values (data_type S) in the address map. These can be results, cycle_length, and analyzer status. The "bad value" is used when certain conditions exist, as in the HCI-H. If the "bad value" is set to zero, no "bad value" processing is done.

2.6 Sample Address Entries

Some sample lines from an address map (the first line is always ignored by the loader):

```
Ref_index, host, tag, desc, data_type, address, anz, app, stream, res, value_type, initvalue,
euhi_address
1, , , , , , sf, 9999          <- scale factor( required)
2, , , , , , bv, 65535        <- "bad value" (required)
3, , , B, 10, , , , DCHG, 0, 1, ,
```

2.7 Example Address Map Configuration

```

4, , ,B,1001,1,1,1, ,SKIP,0,1, ,
5, , ,B,1002,2,1,1, ,SKIP,0,1, ,
6, , ,B,1051,1,1,1, ,SEL,0,1, ,
7, , ,B,1052,2,1,1, ,SEL,0,1, ,
8, , ,B,1101,1,1,1, ,CU,0,1, ,
9, , ,B,1102,2,1,1, ,CU,0,1, ,
10, , ,B,1151,1,1,1, ,ECHG,0,1, ,
11, , ,B,1152,2,1,1, ,ECHG,0,1, ,
12, , ,B,1201,1,1,1, ,DCHG,0,1, ,
13, , ,B,1202,2,1,1, ,DCHG,0,1, ,
14, , ,B,1351,1,1,1, ,CAL,0,1, ,
15, , ,B,1352,2,1,1, ,CAL,0,1, ,
16, , ,B,1667,1,1, , ,SBY,0,1, ,
17, , ,B,1668,2,1, , ,SBY,0,1, ,
18, , ,B,11001,1,1,1, ,RDME,0,1, ,
19, , ,B,11002,2,1,1, ,RDME,0,1, ,
20, , ,S,30001,1,1, , ,AN,0,1, ,
21, , ,S,30002,2,1, , ,AN,0,1, ,
22, , ,S,30255,1,1, , ,CY,0,1, ,
23, , ,S,30256,2,1, , ,CY,0,1, ,
24, , ,S,30509,1,1, , ,DED,0,1, ,
25, , ,S,30510,2,1, , ,DED,0,1, ,
26, , ,S,41001,1,1,1,1,RES, ,1, ,31255
27, , ,S,41002,2,1,1,1,RES, ,1, ,31256
28, , ,E,41255,1,1,1,1,EUHI,1, ,
29, , ,E,41256,2,1,1,1,EUHI,1, ,
30, , ,Q,20001,1,1,1,2,RES, ,1, ,
31, , ,R,70003,2,1,1,2,RES, ,1, ,
    
```

2.7 Example Address Map Configuration

Analyzer 124 has 3 applications:

- Application 1 has stream 1, results 1-3
- Application 2 has stream 4, results 1-3
- Application 3 has stream 1, results 1-3

No control from host is required. The only information from analyzers that is needed by the Host is results and analyzer status.

These addresses do not conform with the HCI-H.

Example Map

host	Da- ta_Type	Address	Anlz	App	Stream	Res	val- ue_typ e	Init val- ue	Slave1	EUHI
1							sf	9999	1	
2							bv	65535	1	

3	R	40001	124	1	1	1	RES	1	0
4	R	70003	124	1	1	2	RES	1	0
5	Q	20005	124	1	1	3	RES	1	0
6	R	40007	124	2	4	1	RES	1	0
7	R	40009	124	2	4	2	RES	1	0
8	R	40011	124	2	4	3	RES	1	0
9	R	40013	124	3	1	1	RES	1	0
10	R	40015	124	3	1	2	RES	1	0
11	R	40017	124	3	1	3	RES	1	0
21	I	30001	124	1			AN	1	
22	I	30002	124	2			AN	1	
23	I	30003	124	3			AN	1	

Note: The results are designated here as 32-bit floating point numbers (data_type Q or R). The second addresses (40002, 70004,...) are not defined in the map, but are present in the driver tables.

2.8 Viewing and Editing the Address Map

Once the address map has been loaded onto the NAU, it may be viewed and edited with the Gas Chromatography Portal from the Analyzer screen under the navigation menu selection Network/Modbus Map or viewed with HMI/Select Analyzer/View Modbus.

The value attribute for GC results can be changed in the table for testing. It contains the pre-scaled version of scaled results and character representations of Boolean values (True or False). Testing for Host commands to the NAU cannot be done by editing the value in the table.

2.9 Configure Analyzers to Transmit Results

For Maxum and MicroSAM analyzers:

1. Use the Gas Chromatograph Portal to add the NAU to the host table of the "sending" analyzers. Use Type "Modbus" and set the Analyzer Id property to the NAU's analyzer id. If the NAU does not appear in the list box, wait until it broadcasts (10-15 minutes).
2. Use the Gas Chromatograph Portal to make entries in the "sending analyzer" result table to indicate which component values to send. Indicate the result values to send by placing an integer value in the TRTVAL to indicate the result # in the address map. EUHI values are set in the result table. The NAU's address map will direct the result to a specific MODBUS address using the anlz, application, stream, and result attributes in the address map table (modbus_addmap).

For Optichrom Analyzers:

1. Set up the results and events for the Optichrom Advance exactly as done for transmitting to an HCI-H.
2. The NAU will need to have a loop, unit defined in order to receive Advance Data Hiway messages. Define the loop and unit in the Gas Chromatograph Portal by selecting the Analyzer View and then the Network menu selection on the Navigation menu. The ADH setup is the "Advance Data Hiway" tab under the Network View. Alternatively, the loop and unit may be set using the HMI/Configure Menu/System Setup screen.

2.10 Special Instructions

Special instructions for Modbus Address Maps:

- Do not duplicate data in the address map. There should be only one address defined for analyzer 1, application 1, Stream 1, result 1. (This differs from the HCIH address map.)
- The address map must contain an ANALYZER STATUS for every transmitting analyzer/application.
- The address map must contain a RESULT for every transmitted component (with EUHI if required).
- If the bad_value is > 0 there will be the normal process of setting values "bad." Otherwise, the component values will be stored as is.
- Prior to Version 3.11, messages from the host must be alternated.
- EUHI values must be >= 1.0.
- Addresses in the 40000s, 20000s, and 70000s all map to the same registers. Do not overlap the addresses. i.e., do not use 40001 and 70001 in the same map.

Host/Analyzer Messages

3.1 Summary of Host/Analyzer Communication

This table summarizes the information types and availability. The topics in this heading provide details for each information type.

Always available to the Host

Result values (Value_type: RESULT)

EUHI values (Value_type:EUHI)

Analyzer/Application status (Value_type: ANALYZERSTATUS)

Available to the Host when defined in the address map

Readme flags (Value_type: rdme)

Database change flags (Value_type: DCHG) (obsolete)

EUHI change flags (Value_type: ECHG) (obsolete)

Stream skip flags(Value_type: SKIPSTREAM)

Dedicated Stream (Value_type: DEDICATEDSTREAM)

Stream active flags (Value_type: CURRENTSTREAM)

Analyzer standby (Value_type: STANDBY)

MapCalibration (Value_type: CALIBRATE)

Cycle Length(Value_type: CYCLELENGTH)

Diread(Value_type: DIREAD)

Airead(value_type: AIREAD)

Alarm(Value_type: ALARM)

Time(Value_type: TIME)

Date(value_type: DATE)

Datetime(value_type: DATETIME)(Version 5.0 replacement for DATE and TIME)

Host Controls sent to Analyzers when addresses are defined

Analyzer control (Value_type: STANDBY)

Calibration control (Value_type: CALIBRATE)

Stream control (Value_type: SELECTSTREAM. SKIPSTREAM)

EUHI change (Value_type: EUHI)

Set clock (Value_types: SCMIN, SCSEC, SCHR, SCDAY, SCMON, SCYR)

Clear Alarms(Value_type: CLEARALARM)

Program Run(Value_type: PROGRAMRUN)

DO set (Value_type: DOSET)

Date(value_type: DATE)

Time (value_type: TIME)

Datetime(value_type: DATETIME)(Version 5.0 replacement for DATE and TIME)

3.2 AIREAD

AIREAD - Valid Addresses: 20001-49999, 70001-79999

3.4 ANALYZERSTATUS

AIREADs can be defined for each Application AI that is defined in the NAU's application 999. It is polled according to the Application AI setup. The value in the result designates the ID from the APPAI table.

Values may be R or Q type. These require 2 consecutive registers (prior to Version 5.0 these were not allowed to cross block boundaries, which are every 1000 registers):

Examples registers of consecutive AIREAD values:

40001, 40003

41999, 42000 (Not allowed prior to Version 5.0, since it crosses the block boundary).

3.3 ALARM

ALARM -

Valid Addresses: 30001-49999

The integer (data_type I) can be defined for each analyzer, analyzer/application, and analyzer/application/stream. It contains the first alarm code for fault alarms.

Valid Addresses: 00001-19999

The boolean (data_type B) can be defined for each analyzer, analyzer/application, and analyzer/application/stream. It contains the presence of alarms on the GC.

It is not possible to define a boolean and an integer with the same analyzer, analyzer/application, and analyzer/application/stream.

Note that the ALARM defined for the analyzer corresponds directly with the red LED on the front of the HMI. An integer type value will reflect the contents of the system_control table errors and warnings attributes. This will be the last fault alarm that occurs or warning alarm, if there are no faults.

An ALARM defined for the stream or application will reflect the contents of the curr_error attribute in the stream and application table. These can only be cleared by the completion of a cycle without alarms. This means that the analyzer ALARM can be zero, but the stream and application can reflect an alarm condition.

3.4 ANALYZERSTATUS

ANALYZERSTATUS - Valid Addresses: 30001-49999

This integer or scaled integer (data_type I or S) value is defined for each application and contains the stream status at the end of cycle. The status values can range from 0 to 1000, as follows:

1000 normal, error free operation

9ss warning alarm on stream ss

7ss auto cal or validation running on stream ss

6ss manual cal running on stream ss

5ss Optichrom Advance change test exceeded on stream ss

4ss Optichrom Advance database change on stream ss

3ss Optichrom Advance excessive rate of change test failed on stream ss

2ss Optichrom Advance minimum rate of change test failed on stream ss
100 application has fault alarm this cycle
50 application is out of service, disabled, or in hold
0 application is not transmitting results(timed out)

3.5 CALIBRATE

CALIBRATE -

For Status: Valid Addresses: 0001-9999

This boolean (data_type B) can be defined for each analyzer/application/stream and is set by the analyzer when the application is in manual or auto calibration. See Host operations in the next section for a discussion of Host calibration control.

For Host Control: Valid Addresses: 0001-9999

The Host can direct an application to calibrate or stop calibrate by setting the calibrate flag, defined on each analyzer/application/stream, for any stream in an application. If set by the Host, a message is sent to the analyzer to place the application in auto calibration (1 or True) or stop calibration (0 or False). In either case, if the analyzer is in hold, the command will be ignored. If auto calibration is not enabled for the application, it will be placed in manual calibration. Although each stream can be defined, it is recommended for only one stream (any stream in the application). There may be operational differences for calibration when the MODBUS interface is implemented for Advance Optichrom analyzers.

3.6 CLEARALARM

CLEARALARM - Valid Addresses: 0001-9999

This boolean (data_type B) can be defined for each analyzer, analyzer/application, and analyzer/application/stream. The Analyzer will clear all alarms for the specified analyzer, application, or stream. This flag is not operational for Advance Optichrom. . Note that clearing alarms operates under the same constraints as clearing the alarms directly on the analyzer. If stream or application alarms are cleared, record of them is still kept by the application and stream until a cycle without alarms completes.

3.7 CURRENTSTREAM

CURRENTSTREAM -

Valid Addresses: 0001-9999

This boolean (data_type B) can be defined for each analyzer/application/stream and is set by the analyzer. The flag tells the Host which stream will be the next to report.

Valid Addresses: 30001-49999

In Version 5.0+, CURRENTSTREAM can be defined as Integer type (I). In that case, the stream is zero and the stream number is in the Value.

3.8 CYCLELENGTH

CYCLELENGTH - Valid Addresses: 30001-49999

This integer (data_type S or I) is set by the analyzer/application to be the longest cycletime in the method table for an application, even unused methods. The longest cycle time is the stream purge time + injection_lag + cycle length.

3.9 DATE and Time

DATE, TIME - (Replaced in version 5.0 with DATETIME, see below)

From the Analyzer: Valid Addresses: 20001-49999, 70001-79999

Date and Time values are transmitted from the analyzer and designate the timestamp of the inject time for most analyzer types. It can be designated for analyzer,application,stream or analyzer,application.

Values may be R or Q type. These require 2 consecutive registers and may not cross block boundaries(which are every 1000 registers):

Examples registers of consecutive TIME and DATE values:

40001, 40003

41999, 42000 is not allowed, since it crosses the block boundary.

From the Host: When a DATE or TIME address is set from the Host, it is a command to set the date. It will not appear changed in the Modbus map. Time and Date must be set together.

Values may be R or Q type. These require 2 consecutive registers and may not cross block boundaries(which are every 1000 registers). TIME must come before DATE.addresses.

Examples registers of consecutive TIME and DATE values:

40001, 40003

41999, 42000 is not allowed, since it crosses the block boundary.

DATE is in the format mmddyyyy

TIME is in the format

Version 5.0:

DATETIME is a unique type that designates a character string.

A Datetime may be defined by analyzer,application,stream or analyzer,application. When being used to set the datetime from the Host, it is recommended to use an address with the stream designated as zero so that it does not interfere with the datetimes coming from the stream cycles. You may use a single Datetime for all streams and for the Host to set the datetime.

3.10 DCHG

DCHG - Valid Addresses: 0001-9999

This boolean (data_type B) can be defined with or without analyzer/application/stream and is set when the scale factor changes. The flag is not reset when read by the Host. If defined for stream 0, it is a summary flag for all streams. If stream 0 entry is set to 0, the flag will set all stream DCHG flags for the application to 0.

3.11 DEDICATEDSTREAM

DEDICATEDSTREAM - Valid Addresses: 30001-49999

This integer (data_type S or I) can be defined for an analyzer/application and is set by the analyzer. It tells the Host if a stream is running on a dedicated basis (Always). If the value is 0, there is no dedicated stream.

3.12 DIREAD

DIREAD - Valid Addresses: 0001-19999

This boolean (data_type B) can be defined for each Application DI that is defined in the NAU's application 999. It is polled according to the Application DI setup. The value in the result field designates the ID from the APPDI table.

3.13 DOREAD

DOREAD - Valid Addresses: 0001-19999

This boolean (data_type B) can be defined for each Application DO that is defined in the NAU's application 999. It is polled according to the Application DO setup(sys_do table has the scanrate). The value in the result field designates the ID from the APPDO table.

3.14 DOSET

DOSET - Valid Addresses: 0001-9999

This boolean (data_type B) can be defined for each Application DO that is defined in the NAU's application 999. It can be set by the host to set the DO in the "on" position for the analyzer.

3.15 ECHG

ECHG - Valid Addresses: 0001-9999

This boolean (data_type B) can be defined for each analyzer/application/stream and is set when the EUHI is changed from the analyzer or from host. Resetting is similar to the RDME.

3.19 RESULT

If defined for stream 0, it is a summary flag for all streams in the application. If stream 0 entry is set to 0, the flag will set all stream ECHG flags for the application to 0.

3.16 EUHI

EUHI -

From analyzer: Valid Addresses: 30001-49999

This is the full scale value for each scaled result. It is set for each transmitted result in the analyzer's result table. The default value is 100. It is sent to the Host as a 16-bit floating point number, as in the HCI-H. The value is useful to the host to verify the full scale value used. Additionally, the EUHI can be set by the Host to force synchronization and override any EUHI set manually on the analyzer. EUHI values must be ≥ 1.0 .

From Host: Valid Addresses: 40001-49999

The EUHI value for any result can be set by the Host in 16-bit floating point format. See the section on the EUHI format. Version 3.0 requires that EUHI values be greater than 1.0.

3.17 PROGRAMRUN

PROGRAMRUN - Valid Addresses: 40001-49999

Registers can be defined for each analyzer/application/stream. The Analyzer runs the event number that is indicated in the register for the designated stream

3.18 RDME

RDME - Valid Addresses: 0001-19999

This boolean (data_type B) can be defined for each analyzer/application/stream. It is set when the analyzer transmits results for a stream. The flag tells the Host that there is new result data available. For versions before 3.11, the flag automatically resets to zero after being read by the Host after 3 seconds. These flags are stored in blocks of 1000 (1-1000, 1001-2000...). The flag is reset after 20 seconds, regardless of host read. Each flag is reset independently. The 20 second default may be changed by setting the write_offset value in the Modbus_addmap table.

Zeros for stream and application are allowed. This would allow all streams to share the same RDME.

3.19 RESULT

RESULT - Valid Addresses: 20001-49999, 70001-79999

Expressed as a fraction of full scale (data_type S) or as a 32-bit float (data_type R). Designated for transmission by the analyzer from the result table or from the Optichrom Advance analyzer.

A "bad value" is used, when defined in the map, when a condition on the analyzer suggests that the values are not current or good. These conditions are:

- the result exceeds the EUHI
- the EUHI has changed
- analyzer status is < 500 (in fault alarm, out of service, or application time limit has expired)
- the stream is disabled.

Values may be R or Q type. These require 2 consecutive registers and, prior to Version 5.0, may not cross block boundaries (which are every 1000 registers):

Examples registers of consecutive RESULT values:

40001, 40003

41999, 42000 is not allowed prior to Version 5.0, since it crosses the block boundary.

Zeros for stream and application are allowed. This would allow all streams to share the same RESULT addresses.

3.20 SCMIN, SCSEC, SCHR, SCDAY, SCMON, SCYR

SCMIN, SCSEC, SCHR, SCDAY, SCMON, SCYR - Valid Addresses: 40001-49999

By writing to the appropriate locations, the Host can set the date and time of day on the Maxum Network Access Unit (NAU). Note: The SCYR address must be the last address written to. These addresses are not kept current by the NAU.

If the NAU has designated another system as its time server, setting from the Host has no long term effect and will be overwritten by the time server. If there is no time server for the NAU, this date and time will be allowed to remain but will have no effect on the analyzers unless each analyzer has its time server set to the NAU's IP address.

3.21 SELECTSTREAM

SELECTSTREAM - Valid Addresses: 0001-9999

This boolean (data_type B), defined on each analyzer/application/stream, can be set by the Host to cause a Force Always condition (1 or True) or a Resume sequence (0 or False) on the stream. The skipstream flag can be set to control stream enable/disable, as described in the help topic for skipstream.

3.22 SKIPSTREAM

SKIPSTREAM - Valid Addresses: 0001-9999

This boolean (data_type B) can be defined for each analyzer/application/stream. It is set by the analyzer and tells the Host if the stream is enabled (0 or False) or disabled (1 or True). If set by the Host, this flag causes a message to be sent to the analyzer to disable or enable the stream.

3.23 STANDBY

STANDBY -

From Analyzer: Valid Addresses: 0001-9999

This boolean (data_type B) can be defined for each analyzer/application and is set by the analyzer. The flag tells the host if the application is in hold, out of service, or disabled.

From DCS: Valid Addresses: 0001-9999

The Host can direct an application to start or stop running. If set by the Host, a message is sent to the analyzer to place the application in hold (1 or True) or run (0 or False).

Modbus Protocol Reference

4.1 Protocol Formats

In the Modbus protocol there are two formats for sending messages, Modbus RTU and Modbus ASCII (not supported). The message content is the same in each format. Both formats have checksums, but the checksum methods are different. All of the Modbus devices on a communications link must use the same message format and the same communication parameters (e.g. baud rate).

4.2 RTU vs. ASCII vs. TCP

Modbus RTU is more efficient than Modbus ASCII, but has stringent timing restrictions that are difficult to comply with when implemented on a multitasking computer, especially using languages such as FORTRAN. For this reason, many Modbus interfaces either deviates from the Modicon specifications regarding Modbus RTU timeouts or else implement the Modbus ASCII format. The Maxum family of products do not support Modbus ASCII.

Modbus TCP utilizes standard TCP/IP messaging to transmit Modbus data. Modbus TCP is distinct from Modbus over TCP. Modbus over TCP takes an entire Modbus RTU message and encapsulates it within a TCP/IP message. Modbus TCP encapsulates only the data portion of the Modbus message. Starting in software version 5.0.7, the Maxum family of products support Modbus TCP. The Maxum family of products do not support Modbus over TCP.

4.3 Modicon Types

In some of the Modicon programming languages, data is viewed as being one of seven types:

- Type 0 digital output (can be read and written)
- Type 1 digital input (can only be read)
- Type 2 digital value representing internal PLC status (exception)
- Type 3 analog input (can only be read)
- Type 4 analog output (can be read or written)
- Type 5 analog value representing internal PLC value (event)
- Type 6 file of 16-bit registers

In Modbus terminology, types 0, 1, 3, 4, and 6 are general-purpose, but type 6 is implemented less frequently than types 0, 1, 3, and 4. Types 2 and 5 are not referred too directly. The type 2 and type 5 data varies between PLC models and can be accessed only through Modbus functions that are often specific to certain PLC models.

4.4 IEEE 32 Bit Float Format

Although the Modbus standard, defined by Modicon, does not specify how to send values other than digitals or 16-bit 2's complement integers, a frequent practice is to use the Modbus protocol to other 16-bit values or to send 32-bit and 64-bit values as pairs and quadruples of registers. For 32-bit and 64-bit values, the most significant 16-bit portion is sent as the first register and the least significant 16-bit portion is sent as the last register. 32-bit IEEE floating point format values are the most common deviation from the Modicon standard but sometimes the values of 32-bit integers and 64-bit IEEE floats are transmitted this way.

The Modbus data model views each type of data as belonging to a separate data table. The conventional notation for addresses specifies which table and the offset within the table. The original ranges of the four main data types are:

- 00001 to 09999 for digital outputs (read and write) meaning table 0 with offsets 1 to 9999
- 10001 to 19999 for digital inputs (read only) meaning table 1 with offsets 1 to 9999
- 30001 to 39999 for analog inputs (read only) meaning table 3 with offsets 1 to 9999
- 40001 to 49999 for analog outputs (read and write) meaning table 4 with offsets 1 to 9999

Extended addresses may be supported in the future.

4.4 IEEE 32 Bit Float Format

The Institute of Electrical and Electronics Engineers has published a standard for floating point formats on computers. The 32-bit version will be supported by the Modbus. The 32-bit values will be packed into pairs of consecutive Modbus registers (16-bit words). The most significant 16 of the 32 bits will be placed into the first register (lower address) and the less significant 16 bits will be placed into the second register (higher address). The most significant bit of the 32 bits will be the most significant bit of the first register. The least significant bit of the 32 bits will be the least significant bit of the second register. The host computer (Modbus master) must extract the 32-bit floating point value from the registers in the Modbus message. 32-bit floating point values may be mixed with 16-bit values in the same Modbus message. The Modbus Address Map defines which Modbus registers hold which type of data. The Modbus supports two floating point formats, one is the standard(data_type R), as described below, and the other is the same format with the words swapped(data_type Q).

NOTE: sending 32-bit IEEE floats using pairs of Modbus registers has become fairly common.

The most significant bit of the IEEE 32-bit float is the sign bit, 0 for a positive number and 1 for a negative number. The next 8 bits are the exponent. If the exponent is 0 (zero), the mantissa represents either zero or a special type of value called Not A Number (NaN) indicating values that cannot be properly represented in the format. NAN numbers are a special case that should not occur in the Siemens system. If the exponent, mantissa, and sign bit are all zero, the value is zero. If the exponent is not zero, the exponent represents a power of two with a bias of 127 and the mantissa represents the fractional part of a value between 1.0 inclusive and 2.0 exclusive. If the exponent is not zero, the value is given as:

$$\text{Value} = (-1)\text{SIGN} * (2.0)(\text{EXPONENT} - 127) * (1.0 + (\text{MANTISSA} / 8388608))$$

S	Exponent								Mantissa																																
3	3	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

Byte 0 (Most Significant)	Byte 1	Byte 2	Byte 3 (Least Significant)
More Significant 16-Bit Word (Word 0)		Less Significant 16-Bit Word (Word 1)	

For example, the Master reads two floating point values from the registers 48601 through 48604, where the slave is identified as slave 65 (0x41). Translating this into a Modbus command, this would be a request to read 4 registers starting at offset 8600 in table 4 of slave 65:

Slave#	Function Code	Data Start		Quantity	
		High	Low	High	Low
41	03	21	98	00	04

The Modbus will respond with a message sending the 4 16-bit words which contain the 2 32-bit values:

Slave#	Function Code	Byte Count	Word 0		Word 1		Word 2		Word 3	
			High	Low	High	Low	High	Low	High	Low
41	03	08	43	1D	66	66	BF	63	D7	0A

In this example, the first value is 157.4 (0x431D6666) and the second value is -0.89 (0xBF63D70A).

4.5 EUHI 16 Bit Float Format

Description

The Optichrom Advance systems HOST Computer Communications Interface – HIWAY (HCI-H) used 16-bit integers to represent the values from analyzer data as scaled fractions of the maximum expected value. The maximum expected value is called the Engineering Unit High (EUHI). The content of the 16-bit integer word is given by:

$$\text{Scaled Result} = \text{Round}(\text{ScaleFactor} * \text{Result value} / \text{EUHI})$$

For example, if the scaling factor is 9999 and the current value is 35.0% of an expected maximum of 50.0%, $9999 * 35.0 / 50.0 = 6999.3$ which would be rounded to 6999 (nearest integer) and stored in the 16-bit word.

The ScaleFactor is normally 999, 4095, 9999, or 65534 but can be configured to any other value between 1 and 65534. A ScaleFactor of 65535 is not allowed because 65535 is the BAD_VALUE (bad quality) indicator. The ScaleFactor must be configured in both the host system and the NAU.

The host system obtains the current value by making this conversion:

$$\text{ResultValue} = \text{EUHI} * \text{Scaled Result} / \text{ScaleFactor}$$

The EUHI is represented using a special 16-bit floating point format derived from the IEEE 32-bit floating point format. The sign bit is still the significant bit (0 for positive, 1 for negative). A 6-bit exponent (bias 31) follows the sign bit. The mantissa occupies the 9 least significant bits. The value 0x0000 represents 0.0. An exponent value of zero with a nonzero mantissa is similar to the Not A Number (NaN) values of the IEEE 32-bit floating point format. Otherwise, the exponent should have a value between 1 and 62, representing powers of 2 between -30 and

4.6 16-Bit Conversion Routines

31 inclusive. An exponent value of 63 is reserved as an indicator of a 'Bad Value'. The EUHI format is only used to represent EUHI values (maximum values):

$$EUHI = (-1)^{SIGN} * (2.0)^{(EXPONENT - 31)} * (1.0 + (MANTISSA / 512))$$

S		Exponent						Mantissa							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte 0 (Most Significant)							Byte 1 (Less Significant)								
16-Bit Word															

4.6 16-Bit Conversion Routines

The following are conversion routines for this format, written in C. The information can be used by host computer software engineers for setting the EUHI from the host or translating the EUHI reported by the NAU. These routines are valid for EUHI values greater than 1.0.

```

int to_euhi(float input);
{
    int output;
    int man;
    int exp;
    float temp;
    for (exp = 0; ; ++exp)
    {
        if ((1 << exp) > input)
            break;
    }
    --exp; /* -1 to compensate for going too far */
    temp = input - (1 << exp);
    man = (int)(temp * (512 / (1 << exp)));
    output = ((exp + 31) << 9) + man;
    return output;
}

/* Convert number from EUHI */
float from_euhi(int input)
{
    float output;
    int man;
    int exp;
    float temp;
    exp = (input >> 9) - 31;
    man = input & 0x1FFF;
    temp = man;
    output = temp / (float)(1 << (9 - exp));
    output += 1 << exp;
    return output;
}

```


4.7 Communication Errors

In both message formats, the message will be discarded if one of these communication errors is detected:

- Received checksum differs from the checksum computed for the message;
- Length of received message differs from length determined for that function;
- If a slave receives a message with a function code of 0 (zero) or between 128 and 255;
- If the master receives a message with a function code of 0;
- Character parity error if either odd or even parity is configured.

As a diagnostic tool, Modbus devices usually maintain message counters that record the number of messages received and the number discarded due to communication errors.

If a slave detects a communication error in a command, the slave discards the message and does not respond to the master. If the master detects a communication error in a response, the master discards the message and retries sending the command. The master will also retry sending a command if an expected response is not received within a configured timeout interval. Master devices may be configured to alert an operator or execute a diagnostic sequence if sending a command fails after a number of retries.

4.8 Implementation of Modbus Protocol

Supported Functions

The Siemens GCs will support the Modbus RTU message format and a subset of Modbus functions and error codes. The Siemens Modbus will extend this subset in two ways:

1. Use of the IEEE 32-bit floating point format (with or without swapped words), packing values into pairs of consecutive Modbus registers;
2. Use of a 16-bit floating point format (EUHI), placing values into Modbus registers for HCI-H compatibility;

Function Codes

These Modbus functions will be supported:

Function Code	Description (not in MODBUS Terminology)
1 (0x01)	Read N booleans from Table 0
2 (0x02)	Read N booleans from Table 1
3 (0x03)	Read N registers from Table 4
4 (0x04)	Read N registers from Table 3
5 (0x05)	Write 1 boolean to Table 0
6 (0x06)	Write 1 register to Table 4
8 (0x08)	Loopback Diagnostic Test (Diagnostic Codes 0, 10, 11, 12, 13)
15 (0x0f)	Write N booleans to Table 0
16 (0x10)	Write N registers to Table 4

Exception Codes

4.10 Message Length

The Modbus will send the following exception response codes:

Exception Code	Cause
01	ILLEGAL FUNCTION (for Function Codes 00, 07, 09 to 14, 17 to 101, 102, and 103 to 127)
02	ILLEGAL DATA ADDRESS (possible for Function Codes 01 to 06, 15, and 16)
03	ILLEGAL DATA VALUE (possible for Function Codes 05 and 08)

4.9 Modbus Message Format

The Modbus protocol is based on polling. One master device polls one or more slave devices. The master sends commands (queries) to the slaves. The slaves wait for commands from the master. If a command is directed to a particular Modbus slave, the slave sends a response. Some commands can also be broadcast to all slaves at once. The Modbus implementation does not support broadcast messages.

4.10 Message Length

The message lengths are specified for the bytes in the message and ignore checksum. Modbus RTU requires 2 additional bytes for the checksum.

Function Code	Length of Command	Length of Response
0	Communication Error	
1	6	3 + value of third byte
2	6	3 + value of third byte
3	6	3 + value of third byte
4	6	3 + value of third byte
5	6	6
6	6	6
7	2	3
8	6	6
9	2 + value of fifth byte	2 + value of fifth byte
10	2	2 + value of fifth byte
11	2	3 + value of third byte
12	2	3 + value of third byte
13	3 + value of third byte	3 + value of third byte
14	2	3 + value of third byte
15	7 + value of seventh byte	6
16	7 + value of seventh byte	6
17	2	3 + value of third byte
18	3 + value of third byte	3 + value of third byte
19	3 + value of third byte	3 + value of third byte
20	2 + value of third byte	2 + value of third byte

21	2 + value of third byte	2 + value of third byte
22-126	3 + value of third byte	3 + value of third byte
127	2	2
128-255	Communication Error	3

4.11 Message Content

The message content is always:

Slave#	Function Code	String_of_Data_Bytes
1 byte	1 byte	N bytes

Each message begins with a one byte slave identifier (Slave#). Slaves are configured with identifiers between 1 and 255 inclusive. Slave identifier 0 (zero) is used in the commands broadcast to all slaves. Broadcast messages are not supported in the Modbus. Note: Modicon documentation specifies 1 to 247 for slave identifiers, reserving 248 through 255 for special equipment, but other vendors frequently allow 1 to 255 for slave identifiers.

The second byte of the message indicates the function code (FunctionCode). For commands, this byte has a value between 0 and 127 inclusive. For responses, the function code values range from 1 to 127 if there was no error and from 128 to 255 if there was an error. Slaves indicate errors by sending a response with 128 added to the function code and sending an error code as the String_Of_Data_Bytes.

The String_Of_Data_Bytes has a length that depends upon the function code and whether the Modbus Master is sending the message (Command) or a Modbus Slave is sending the message (Response). For some function codes, the length of the String_Of_Data_Bytes can vary. If the length can vary, there is always one byte, which indicates the number of bytes, 1 to 255, in the variable portion.

4.12 Digital Values

Modbus messages that transmit a string of digital values pack these into a sequence of bytes. The first 8 digitals are packed into the first byte, the next 8 digitals are packed into the second byte, and so on. Digital values are packed into a byte using the least significant available bit. The value of the first digital value is given by the least significant bit of the first byte. If there are not enough digital values to fill the last data byte, the unused bits of that byte will be in the most significant bit positions and have a value of 0.

For an example of packing bits into bytes, assume the following sequence of 19 digital values will be packed into 3 consecutive bytes:

0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, and 1,

Then: the first eight bits (0,0,1,0,1,1,1,1) would be packed into the first byte which would have the value 0xF4 (11110100), the next eight bits (1,0,0,0,0,0,1,0) would be packed into the second byte which would have the value 0x41 (01000001), and the last three bits (1,0,1) would

4.15 Checksum and Communication Errors

be packed into the third byte which would have the value 0x05 (0000101) because the most significant 5 bits of the third byte would be set to zero (bit positions shown below)

Data Byte 0								Data Byte 1								Data Byte 2							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	1

One Modbus function sets the value of a single digital (Function 05 – Force Single Coil). This particular function represents the value of ON by 0xFF00 (65023 decimal) and the value of OFF by 0x0000 (zero).

Within a Modbus message, the values of 16-bit quantities are sent with the more significant byte transmitted before the less significant byte. Note that this is opposite of the way Modbus RTU transmits the CRC-16 checksum.

4.13 RTU Checksum

Modbus RTU uses a CRC-16 (cyclic redundancy check) checksum, an equivalent to doing a polynomial division of the message in modulo-65536 (i.e., keep the remainder and discard the quotient). The algorithm usually does this one bit at a time. However, modulo arithmetic allows this to be done one byte at a time, by precomputing a table of all 256 of the possible CRC-16 checksums for a single-byte message.

To compute the 16-bit CRC-16 checksum for a message:

1. Initialize the checksum to 0xFFFF (65535 in decimal);
2. For each byte in the message, starting with the first, do:
 - Exclusive OR the message byte with the less significant byte of the checksum;
 - Use the result of (2.a) as an index into the precomputed CRC-16 lookup table;
 - Exclusive OR the result of (2.b) with the more significant byte of the checksum;
 - Replace the current checksum with the result of (2.c);

4.14 Basic Data Types

The Modbus protocol has two basic data types, digital and analog. Digital values are either 0 (zero) or 1 (one). Analog values are usually 16-bit 2’s complement integers, giving a range from –32768 to 32767. The Modbus protocol also uses 16-bit positive integers with a range of 0 to 65535 (for example, Modbus addresses). Digital values are often referred to as coils. Analog values and other 16-bit values are often referred to as registers.

4.15 Checksum and Communication Errors

When a message is received, the CRC-16 checksum of the message is computed and compared with the received CRC-16 checksum. If the two checksums match, there was probably no communication error.

4.16 CRC Lookup Table

Use the following steps to create the CRC-16 lookup table. This is only one of many ways to compute CRC-16 checksums.

- (1) For each of the values 0 to 255, do:
 - (1.a) Initialize the checksum to 0xFFFF (65535 in decimal);
 - (1.b) Exclusive OR the less significant byte of the checksum with the value between 0 and 255;
 - (1.c) Replace the less significant byte of the checksum with the result of (1.b);
 - (1.d) Do the following 8 times:
 - (1.d-1) If the least significant bit of (1.c) is 1 then:
 - (1.d-1.a) Right shift the checksum one bit;
 - (1.d-1.b) Exclusive OR the result of (1.d-1) with 0xA001 (40961 in decimal);
 - (1.d-1.c) Replace the checksum with result of (1.d-2);
 - (1.d-2) Else:
 - (1.d-2.a) Right shift the checksum one bit;
 - (1.d-2.b) Replace the checksum with the result of (1.e-1);
 - (1.e) Place the result of (1.d) in the table at the index specified by the value between 0 and 255.

4.17 Character Transmission

Modbus protocol uses normal RS-232C, RS-422 (not supported), and RS-485 asynchronous character transmission. A start bit is sent before the data bits, then the data bits are transmitted with the least significant bit first, then an optional parity bit may be sent, and finally one or two stop bits are sent.

In Modbus RTU, the binary values of the message and checksum are transmitted as characters. The checksum requires two bytes, which are transmitted with less significant byte first, immediately after the last byte of the message is transmitted. Characters must be transmitted as 8 data bits. Character parity can be odd, even, or none. Either 1 or 2 stop bits can be used.

In Modbus RTU, a pause is inserted between messages to indicate the start and end of a message. Modicon's Modbus specification requires a pause equal to or greater than the time required to transmit 3.5 characters at the configured baud rate. Shorter pauses are accepted between characters in the same message. However, many Modbus RTU interfaces are implemented on multiprocessing computers and are unable to detect such a small time interval, especially at higher baud rates. For these systems, a larger pause is specified (for example, minimum of 500ms inserted between consecutive messages).

The following table assumes: 8-bit data, 1 start bit, 1 stop bit, and no parity.

Baud Rate (bits per second)	3.5 Character Pause (milliseconds)
110	318.2
300	116.7
1200	29.2

4800	7.29
9600	3.65
19200	1.823
38400	0.911
57600	0.608
112500	0.304

4.18 Function Codes

Modicon has defined many Modbus Function Codes, most of which are specialized for PLC configuration and some are only used with specific models of PLC's. Most vendors only implement a subset of the Modbus functions.

The following functions are almost always supported:

Function 01	Read Coil Status	reads a string of digital outputs (table 0)
Function 02	Read Input Status	reads a string of digital inputs (table 1)
Function 03	Read Holding Registers	reads a string of analog outputs (table 4)
Function 04	Read Input Registers	reads a string of analog inputs (table 3)

The following functions are almost as common:

Function 05	Force Single Coil	writes a value to a single digital output (table 0)
Function 06	Preset Single Register	writes a value to a single analog output (table 4)

The following functions are also very common:

Function 15	Force Multiple Coils	writes values to a string of digital outputs (table 0)
Function 16	Preset Multiple Registers	writes values to a string of analog outputs (table 4)

Note: At least one of Function 05 and Function 15 will be implemented. Usually both are. At least one of Function 06 and Function 16 will be implemented. Usually both are.

Less common functions are:

Function 08	Loopback Diagnostic Test	diagnostic test message sent to test communications
Function 11	Fetch Event Counter Com- munications	check slave's communications counter

The other Modbus functions tend to be specific to models of Modicon PLC's and are not described here.

These Modbus functions will be supported in the Maxum/MicroSAM:

Function Code	Description (not in MODBUS Terminology)
1 (0x01)	Read N booleans from Table 0
2 (0x02)	Read N booleans from Table 1
3 (0x03)	Read N registers from Table 4
4 (0x04)	Read N registers from Table 3
5 (0x05)	Write 1 boolean to Table 0
6 (0x06)	Write 1 register to Table 4
8 (0x08)	Loopback Diagnostic Test (Diagnostic Codes 0, 10, 11, 12, 13)
15 (0x0f)	Write N booleans to Table 0
16 (0x10)	Write N registers to Table 4

4.19 Exception Response Codes

The Modbus slave can send the following exception response codes:

Exception Code	Cause
01	ILLEGAL FUNCTION (for Function Codes 00, 07, 09 to 14, 17 to 101, 102, and 103 to 127)
02	ILLEGAL DATA ADDRESS (possible for Function Codes 01 to 08, 15, and 16)
03	ILLEGAL DATA VALUE (possible for Function Codes 05 and 08)

4.20 Error Responses

If a slave receives a command with the slave's identifier and with a function code that the slave doesn't support, the slave will send an error response indicating that an ILLEGAL FUNCTION was received. The slave will send a 3-byte error response with its identity, the function code summed with 128, and the ILLEGAL FUNCTION error code.

For example, if slave 17 (0x11) receives a request for function 98 (0x62), slave 17 will respond with identity 0x11, function code 0xE2 (128+98), and error code 0x01 (ILLEGAL FUNCTION):

Slave#	Function Code	Error Code
11	E2	01

4.21 Function Reference

4.21.1 Function 01 - Read Coil

Master Command

The master sends a command requesting the values of a number of digital outputs (table 0), specifying the initial address to read and the number of locations to read. For example, the master might request a read of the 37 coils 00020 through 00056 from slave 17. Translating the values into hexadecimal, this is a request that slave 0x11 read 0x0025 digitals starting at address 0x0013 (corresponds to table offset 0020 of table 0):

Slave#	Function Code	Data Start		Quantity	
		High	Low	High	Low
11	01	00	13	00	25

Slave Response

If the slave can process the command without error, the slave will respond with its identifier, the function code, a count specifying the number of data bytes required to pack the digitals, and the string of data bytes. For example:

Slave#	Function Code	Byte Count	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4
11	01	05	CD	6B	B2	0E	1B

In this example,

- 0xCD (11001101) indicates that coils 00020, 00022, 00023, 00026, and 00027 are on.
- 0x6B (01101011) indicates that coils 00028, 00029, 00031, 00033, and 00034 are on.
- 0xB2 (10110010) indicates that coils 00037, 00040, 00041, and 00043 are on.
- 0x0E (00001110) indicates that coils 00045, 00046, and 00047 are on.
- 0x1B (00011011) indicates that coils 00052, 00053, 00055, and 00056 are on.
- All of the other coils are off (value of zero).

Note: the three most significant bits of data byte 4 (fifth byte) are all unused and are therefore set to zero.

Possible Errors

Possible errors that could occur are:

- The specified starting address is outside of the configured range for table 0.
- The combination of specified starting address and number of digitals (quantity) would result in an address outside of the configured range for table 0.
- The specified number of digitals (quantity) is too large to fit into a response. (A response can hold a maximum of 255 data bytes or 2040 digitals.)

Error Response

Some devices limit the response length to 250 data bytes (2000 digitals). If one of the first two errors occurs, an error response will be sent indicating that there is an ILLEGAL DATA ADDRESS. If the third error occurs, an error response will be sent indicating that there is an ILLEGAL DATA VALUE.

Using the example above slave 17 would respond to a command which had an illegal address or an illegal combination of address and number of digitals (quantity) with:

Slave#	Function Code	Error Code
11	81	02

Slave 17 would respond to a command specifying more than the allowable number of digitals (quantity) with:

Slave#	Function Code	Error Code
11	81	03

4.21.2 Function 02 - Read Input**Master Command**

The master sends a command requesting the values of a number of digital inputs (table 1) that specifies the initial address to read and the number of locations to read. For example, the master might request a read of the 22 inputs 10197 through 10218 from slave 17. Translating the values into hexadecimal, this is a request that slave 0x11 read 0x0016 digitals starting at address 0x00C4 (corresponds to table offset 0197 of table 1):

Slave#	Function Code	Data Start		Quantity	
		High	Low	High	Low
11	02	00	C4	00	16

Slave Response

If the slave can process the command without error, the slave will respond with its identifier, the function code, a count specifying the number of data bytes required to pack the digitals, and the string of data bytes. For example:

Slave#	Function Code	Byte Count	Data Byte 0	Data Byte 1	Data Byte 2
11	02	03	AC	DB	35

In this example,

- 0xAC (10101100) indicates that inputs 10199, 10200, 10202, and 10204 are on.
- 0xDB (11011011) indicates that inputs 10205, 10206, 10208, 10209, 10211, and 10212 are on.
- 0x35 (00110101) indicates that inputs 10213, 10215, 10217, and 10218 are on.
- All of the other inputs are off (value of zero).

Note: the two most significant bits of data byte 2 (third byte) are all unused and are therefore set to zero.

Error Response

Similar to a Function Code 01, an ILLEGAL DATA ADDRESS error response will be sent if the starting address or combination of starting address and number of digitals exceeds the configured range of table 1 (digital inputs).

An ILLEGAL DATA VALUE error response will be sent if the number of digitals exceeds 2040, the limit that can be packed into 255 data bytes for a response.

Note: some devices will limit the response length to 250 data bytes (2000 digitals). However, the function code in the error response, will be 130 (0x82) indicating that the error occurred on a command with Function Code 02.

4.21.3 Function 03 - Read Output

Master Command

The master sends a command requesting the values of a number of output registers (table 4), specifying the initial address to read and the number of locations to read. For example, the master might request a read of the 3 registers 40108 through 40110 from slave 17. Translating the values into hexadecimal, this is a request that slave 0x11 read 0x0003 analogs starting at address 0x006B (corresponds to table offset 0108 of table 4):

Slave#	Function Code	Data Start		Quantity	
		High	Low	High	Low
11	03	00	6B	00	03

Slave Response

If the slave can process the command without error, the slave will respond with its identifier, the function code, a count specifying the number of data bytes required for the analogs, and the string of data bytes. For example:

Slave#	Function Code	Byte Count	Word 0		Word 1		Word 2	
			High	Low	High	Low	High	Low
11	03	06	02	2B	00	00	00	64

In this example, registers 40108, 40109, and 40110 have the values 555 (0x022B), 0 (0x0000), and 100 (0x0064) respectively.

Error Response

Similar to Function Code 01, an ILLEGAL DATA ADDRESS error response will be sent if the starting address or combination of starting address and number of analogs exceeds the configured range of analog outputs (table 4).

An ILLEGAL DATA VALUE error response will be sent if the number of registers exceeds 127 which is the limit that can be packed into 254 data bytes for a response.

Note: Some devices limit the response length to 250 data bytes (125 registers). However, the function code in the error response will be 131 (0x83), indicating that the error occurred on a command with Function Code 03.

4.21.4 Function 04 - Read Input

Master Command

The master sends a command requesting the values of a number of input registers (table 3), specifying the initial address to read and the number of locations to read.

For example, the master might request a read of the register 30009 from slave 17. Translating the values into hexadecimal, this is a request that slave 0x11 read 0x0001 analogs starting at address 0x0008 (corresponds to table offset 0009 of table 3):

Slave#	Function Code	Data Start		Quantity	
		High	Low	High	Low
11	04	00	08	00	01

Slave Response

If the slave can process the command without error, the slave will respond with its identifier, the function code, a count specifying the number of data bytes required for the analogs, and the string of data bytes. For example:

Slave#	Function Code	Byte Count	Word 0	
			High	Low
11	03	02	00	00

In this example, register 30009 has the value zero (0x0000).

Error Response

Similar to Function Code 01, an ILLEGAL DATA ADDRESS error response will be sent if the starting address or combination of starting address and number of analogs exceeds the configured range of table 3 (analog inputs).

An ILLEGAL DATA VALUE error response will be sent if the number of registers exceeds 127, the limit that can be packed into 254 data bytes for a response.

Note: Some devices limit the response length to 250 data bytes (125 registers). However, the function code in the error response will be 132 (0x84), indicating that the error occurred on a command with Function Code 04.

4.21.5 Function 05 - Set Single Coil

Master Command

The master sends a command writing the value of one output digital (table 0), specifying the digital's address and the value to write. For example, the master might request that slave 17 turn ON coil 00173. Translating the values into hexadecimal, this is a request that slave 0x11 write 0xFF00 to the coil at 0x00AC (corresponds to table offset 0173 of table 0):

Slave#	Function Code	Data Start		Digital Value	
		High	Low	High	Low
11	05	00	AC	FF	00

Slave Response

If the slave can process the command without error, the slave will echo the command as a response:

Slave#	Function Code	Data Start		Digital Value	
		High	Low	High	Low
11	05	00	AC	FF	00

Possible Error

A possible error that might occur is that, although the address and data value are valid, there is some internal error that prevents the digital value from being written to the specified digital. For instance, the value may be on another device connected over an internal network. If the other device is offline or has security features, which can block writing to that device's digital, this would cause a FAILURE IN ASSOCIATED DEVICE error response.

For example, if slave 17 had not been able to carry out the command because of an internal system problem, the response would have been:

Slave#	Function Code	Error Code
11	85	04

Similar to Function Code 01, an ILLEGAL DATA ADDRESS error response will be sent if the coil's address is outside the configured range of table 0 (digital outputs).

An ILLEGAL DATA VALUE error response will be sent if the digital value specified is neither 0xFF00 (ON) nor 0x0000 (OFF). However, the function code in the error response will be 133 (0x85) indicating that the error occurred on a command with Function Code 05.

4.21.6 Function 08 - Loopback Diagnostic

Master Command

The master sends the Loopback Diagnostics Test command to test communications with a slave device. A 2-byte diagnostic code tells the slave what action is required. Modicon has specified a number of diagnostic codes and reserves some additional codes for proprietary use. Most of the diagnostic codes tell the slave to return a value from an internal communications event counter (e.g., Diagnostic Code 12 requests the Checksum Error Count). The 2-byte data code is followed by a two byte (16-bit) data value.

Diagnostic Codes

Diagnostic Code	Action of Modbus Slave Device
0	Respond by echoing command
1	Reinitialize Modbus interface
10	Clear event counters (reset to zero)
11	Respond with the number of received messages (bus message events)
12	Respond with the number of received checksum errors (checksum error events)
13	Respond with the number of error responses sent (exception events)
14	Respond with the number of responses sent (response events)
15	Respond with the number of commands not responded to (nonresponse events)
16	Respond with the number of NAK responses from attached devices (NAK events)
17	Respond with the number of busy responses from attached devices (busy events)

Diagnostic Codes

Diagnostic Code	Action of Modbus Slave Device
-----------------	-------------------------------

Diagnostic Codes 16 and 17, NAK, and busy event counters respectively are a diagnostic for Modbus slave interfaces that forward values to attached devices perhaps over a proprietary network.

Full definitions of these codes are shown below.

Diagnostic Code 0

Return Query Data, is the most useful. This allows the host system (Modbus master) to test loop communications and verify that the slave device can correctly generate checksum values (CRC-16 for Modbus RTU and LRC for Modbus ASCII). Verifying the checksum requires sending a series of Loopback Diagnostics Test messages with different 16-bit data values. The addressed slave will send a response to this diagnostic code. The response should match the message sent by the master, but a checksum will be generated by the slave and is not just an echo of the command's checksum.

Diagnostic Code 1

Restart Communications Option, tells the slave to reinitialize all serial communications including clearing all event counters (message and error counters). After initialization, the slave will resume waiting to be polled by the master. The slave does not send a response to this diagnostic code. Modicon specifies two data values for this diagnostic code indicating whether the slave should halt or continue when a communications error is detected. A data value of 0x0000 specifies the slave should halt on communications error. A value of 0xFF00 says the slave should continue on communications error (but increment error event counters).

Note: For diagnostics on remote devices in SCADA systems, it is useful to halt the slave device when an error occurs and then read a communications log (Function Code 12). Otherwise, the Modbus implementation should ignore the data values for Diagnostic Code 1 (i.e., accept all values).

Diagnostic Code 10

Clear Counters and Diagnostic Register, is useful if Function Code 11, Fetch Communications Event Counter, is implemented. A Modbus slave increments the event counter for every successful command from the Modbus master (i.e., commands that caused no errors). A Modbus master can send a series of commands and then check the event counter to verify that all were successful. This can be useful in testing a new configuration. The slave sends a response that echoes the command from the master.

Diagnostic Codes 11 through 17

Diagnostic Codes 11, 12, 13, 14, 15, 16, and 17 are useful to allow a host computer system (Modbus master) to track communication statistics on Modbus slaves. The slaves respond with the values of event counters.

Note: For Diagnostic Codes 10 through 17, the master should send a data value of 0 (zero). Unless there is reason to do otherwise, the slave device should ignore the data value (i.e., accept every data value) for these diagnostic codes. For Diagnostic Codes 11 through 17, the slave should respond with the value of the specified event counter. For Diagnostic Code 10, the slave should respond by echoing the command (similar to the response for Diagnostic Code 0).

Error Response

The only error that should occur is that a command with an unsupported diagnostic code is sent. If the slave does not recognize the diagnostic code, it should send an ILLEGAL DATA VALUE error response (see Function Code 01 on page 43) with the Function Code set to 136 (0x88) to indicate that the error occurred on a command with Function Code 08.

Error Example 1

The master sends Diagnostic Code 0 with data value 42295 (0xA537) to slave 17:

Slave#	Function Code	Diagnostic Code		Data Value	
		High	Low	High	Low
11	08	00	00	A5	37

The slave will echo the command as a response:

Slave#	Function Code	Diagnostic Code		Data Value	
		High	Low	High	Low
11	08	00	00	A5	37

Error Example 2

The master sends Diagnostic Code 12 with data value 0 (zero) to slave 17:

Slave#	Function Code	Diagnostic Code		Data Value	
		High	Low	High	Low
11	08	00	0C	00	00

The slave will respond with its identity, the function code, the diagnostic code, and the value of the event counter (count of checksum errors), a value of 3 in this example:

Slave#	Function Code	Diagnostic Code		Data Value	
		High	Low	High	Low
11	08	00	0C	00	03

4.21.7 Function 15 - Set Multiple Coils

Master Command

The master sends a command writing the values to a string of digital outputs (table 0), specifying the starting address, the number of digitals, and the values. The values are packed into bytes. The first byte contains the values of the first eight digitals, the second byte contains the values of the next eight digitals, and so on. The values are packed into the bytes, beginning with the least significant bit of a byte and finishing with the most significant bit of the byte. If there are not enough digitals to fully use all of the bits of the last data byte, the unused bits should be set to zero. There should be at least one digital value in the last data byte.

For example, the master might set the values of the 10 coils 00014 through 00023, turning ON (one) coils 00014, 00016, 00017, 00020, and 00021 while turning OFF (zero) coils 00015, 00018, 00019, 00022, and 00023:

Slave#	Function Code	Data Start		Quantity		Byte Count	Data Byte 0	Data Byte 1
		High	Low	High	Low			
11	0F	00	13	00	0A	02	CD	00

Slave Response

If the slave can process the command without error, the slave will send a response with its identity, the function code, the starting address, and the number of digitals (quantity) whose values were set:

Slave#	Function Code	Data Start		Quantity	
		High	Low	High	Low
11	0F	00	13	00	0A

Error Response

4.21 Function Reference

Similar to Function Code 05, an ILLEGAL DATA ADDRESS error response will be sent if the register’s address is outside the configured range of table 0 (digital outputs) or if the combination of starting address and quantity results in an address outside of the configured range.

A FAILURE IN ASSOCIATED DEVICE error response will be sent if an internal error prevented the command from being carried out.

Normally there would not be an ILLEGAL DATA VALUE response because all values should be acceptable. The function code in the error response will be 143 (0x8F) indicating that the error occurred on a command with Function Code 15.

4.21.8 Function 16 - Set Multiple Registers

Master Command

The master sends a command writing the values to a string of analog outputs (table 4), specifying the starting address, the number of analog values, and the 16-bit values. The values are sent with the more significant byte before the less significant byte.

For example, the master might set the values of the 2 registers 40136 and 40137 to 10 (0x000A) and 258 (0x0102) respectively in slave 17:

Slave#	Function Code	Data Start		Quantity		Byte Count	Word 0		Word 1	
		High	Low	High	Low		High	Low	High	Low
11	10	00	87	00	02	04	00	0A	01	02

Slave Response

If the slave can process the command without error, the slave will send a response with its identity, the function code, the starting address, and the number of analogs (quantity) whose values were set:

Slave#	Function Code	Data Start		Quantity	
		High	Low	High	Low
11	10	00	87	00	02

Error Response

Similar to Function Code 05, an ILLEGAL DATA ADDRESS error response will be sent if the register’s address is outside the configured range of table 4 (analog outputs) or if the combination of starting address and quantity results in an address outside of the configured range.

A FAILURE IN ASSOCIATED DEVICE error response will be sent if an internal error prevented the command from being carried out.

Normally there would not be an ILLEGAL DATA VALUE response because all values should be acceptable. The function code in the error response will be 144 (0x90) indicating that the error occurred on a command with Function Code 16.

Appendix A - Contact Information

A.1 Product Information Availability

Information and instructions to use this product are available. The procedure can be obtained from the following web location:

www.usa.siemens.com/gc-manuals

- For Process Gas Chromatograph Manuals, click on the specific link.
- For MAXUM Edition II and MicroSAM Procedures and Updates, use the browser search function (Ctrl-F) to search for the part number or name of this product.

SIOS - Siemens Industry Online Support

The foundation for product information is the SIOS site at <http://support.industry.siemens.com/cs/products>.

Searching for Manuals

The following address starts a search for Maxum II manuals:

<http://support.industry.siemens.com/cs/products?dtp=Manual>

A.2 Contacts

A.2 Contacts

Register at the Siemens Industry Online Support (SIOS) website:
<https://support.industry.siemens.com>

International	USA
<p>Siemens AG I IA SC PA PM Process Analytics Oestliche Rheinbrueckenstrasse 50 76187 Karlsruhe Germany Web site: www.siemens.com/processanalytics Support Information: https://support.industry.siemens.com</p> <p>Spares Contact your local Siemens sales representative</p> <p>Support Requests www.siemens.com/automation/support-request</p>	<p>Siemens Industry, Inc. 5980 West Sam Houston Parkway North Suite 500 Houston, TX 77041 USA Tel: +1 713 939 7400 Fax: +1 713 939 9050 Email: saasales.industry@siemens.com Web site: www.usa.siemens.com/pa</p> <p>Training Tel: +1 800 448 8224 (USA) Email: saatraining.industry@siemens.com</p> <p>Spares Tel: +1 800 448 8224 (USA) Email: PAspareparts.industry@siemens.com</p> <p>Support Requests www.siemens.com/automation/support-request Tel: +1 800 448 8224 (USA) Email: GCsupport.industry@siemens.com</p>

Singapore	Online Support Request
<p>Siemens Pte. Limited I IA SC Process Analytics 9 Woodlands Terrace Singapore 738434 Web Site: http://www.siemens.com.sg</p>	<p>Web site: www.siemens.com/automation/support-request</p>

Appendix B - Change Log

B.1 December 2019 Changes

1. Added Security information (Page 7) disclaimer.

