AQUISIÇÃO DE ANALISADORES ON LINE UO1 2022

Autores:
Roberto J. C. Stoll
Gustavo Rinaldi
Nuria Carvalho Neves

KEMPETRO
ENGENHARIA

A	SEM COMENTÁRIOS. Enviar cópias certificadas. Prosseguir a Fabricação
B	COM COMENTÁRIOS. Reemitir documento para comentários. Prosseguir fabricação com exceção das partes comentadas
C	COM COMENTÁRIOS. Enviar cópias certificadas. Prosseguir a fabricação
D	REJEITADO. Reenviar para comentários. NÃO PROSSEGUIR a fabricação
E	ACEITE DO CERTIFICADO
F	PARA INFORMAÇÃO
Ass__________ Data ___/___/___	

OS COMENTÁRIOS FEITOS NESTE DOCUMENTO NÃO EXIMEM O FABRICANTE DA RESPONSABILIDADE SOBRE O PROJETO, A FABRICAÇÃO E O DESEMPENHO DO EQUIPAMENTO OU SISTEMA.

RESPONSÁVEL TÉCNICO	RM Braskem Nº BK-BA01-01400-RM-81-00058		PJ Braskem Nº PJ-0601721
Nome Guilherme Notti	Doc. Fornecedor Nº 12564.1/2021		TAG AT-05670
CREA RS 107210	Título Cromatógrafo de Gás - Manual Modbus		Nome
ART			Crachá
Visto			Visto
Data 15/02/23	N° Pedido Braskem 4503083294	N°Item 3211732	Data

Código de Emissão (Finalidade)

PR – Preliminar	CO – Para Comentários	LD – Liberado para Det.	CS – Cancel./Substit.
LH – Liberação com HOLD	CE – Certificado	CP – Conforme Comprado	CA – Cancelado
IN – Para Informação	PP – Para Compra	CC – Conforme Contruído	
PA - Para Aprovação	LE – Liberado para Execução		

	Descrição de revisão								
	Visto / Crachá								
	Nome								
	Data								
	Descrição de revisão								
	Visto / Crachá								
	Nome								
	Data								
	Descrição de revisão								
	Visto / Crachá								
	Nome								
	Data								
0	Descrição de revisão	Para aprovação							
	Visto / Crachá								
	Nome	RS	GR	NN					
	Data	15/02/23	15/02/23	15/02/23					
REV		Autor	Verificação	Aprovação	Aprovações - Interfaces			Aceitação	GQ
		Emissão							Liberação

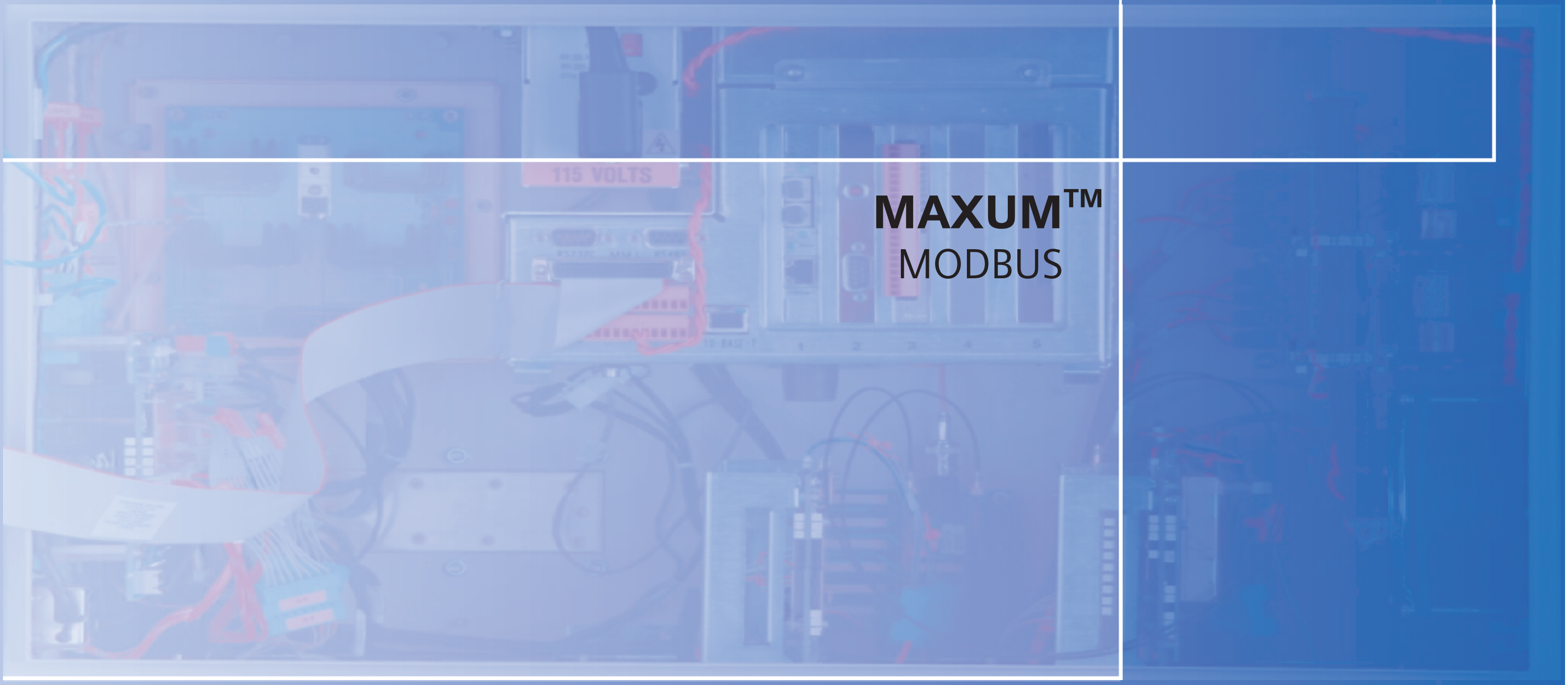
Braskem	Projeto AQUISIÇÃO DE ANALISADORES ON LINE UO1 2022		
	PJ Braskem Nº PJ-0601721	Emissão: 15/02/23	
ISOCELL Soluções em Analítica	RM Braskem Nº BK-BA01-01400-RM-81-00058		
	Doc. Fornecedor Nº 12564.1/2021	Pág.: 1	Rev.: 0
Título: Cromatógrafo de Gás - Manual Modbus		Autor: RS	

Folha para controle de revisões

Notas e alterações ocorridas:

Rev.		1	2	3	4	5	6
Página	Rev.de cada página						
01	*X*						
02							
03							
04							
05							
06							
07							
08							
09							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							
34							
35							
36							
37							
38							
39							
40							
41							
42							
43							
44							
45							
46							
47							
48							
49							
50							
51							
52							
53							
54							
55							
56							
57							
58							
59							
60							
61							
62							
63							
64							
65							
66							
67							
68							
69							
70							
71							
72							
73							
74							
75							
76							
77							
78							
79							
80							
81							
82							
83							
84							
85							
86							
87							
88							
89							
90							
91							
92							
93							
94							
95							
96							
97							
98							
99							
100							
101							
102							

User's Guide 1/2007



MAXUM™ MODBUS

process
GAS CHROMATOGRAPHY

SIEMENS

SIEMENS

Maxum™ MODBUS

Revised Printing – January 2007

User's Guide
2000584-001

Copyright Notice

Trademarks

Table of Contents

Topic	Page
Maxum Modbus Basics	2
Maxum MODBUS Operation	3
System Specifications	4
Component Values	6
Data Validity and Operational Checks	7
Host / Analyzer Communications	9
Maxum MODBUS Address Map	15
Example Map Configurations	22
MODBUS Messages	26
Floating Point Formats	27
Getting Started	31
Formula for Map Size	33
MODBUS Primer	35
MODBUS Data Types	41
MODBUS Functions	43
Function 01 – Read Coil Status	45
Function 02 – Read Input Status	47
Function 03 – Read Output Registers	49
Function 04 – Read Input Registers	50
Function 05 – Force Single Coil	51
Function 06 – Preset Single Register	53
Function 08 – Loopback Diagnostics Test	54
Function 15 – Force Multiple Coils	58
Function 16 – Preset Multiple Registers	59
Maxum Implementation of MODBUS Protocol	60
References	61
Index	Index -1

Maxum MODBUS Basics

Purpose

This manual provides detailed information about the hardware and software protocol required to interface a computer to the Maxum™ system (starting with Maxum Version 3.0). The connection between an external computer and the Maxum is made via a serial, digital communication link. The purpose of the connection is to allow Maxum and Optichrom Advance analyzers to transmit the results of chromatographic analyses to the external computer and to allow the external computer to perform certain control operations of a Maxum or Optichrom Advance analyzer.

Definition of Terms

Term	Definition
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.
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 Maxum system.
Master	MODBUS systems require one master device (host) which sends requests to one or more slave devices.
Maxum	Siemens’ Advance Maxum product line of gas analyzers and network devices.
MODBUS	Communications protocol, defined by Gould, Inc. for the Gould MODICON, that has become a de facto standard for data communications.
NAU	Advance Network Access Unit (NAU) provides general purpose I/O for Advance Maxum systems. Any Maxum unit 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 Maxum product line.
Register	MODICON 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.
Slave	MODBUS device that passively waits for requests from a MODBUS master device.

Maxum MODBUS Operation

Description

This section provides an overview of the Maxum MODBUS operation.

Operation

From Maxum 3.1 Analyzer

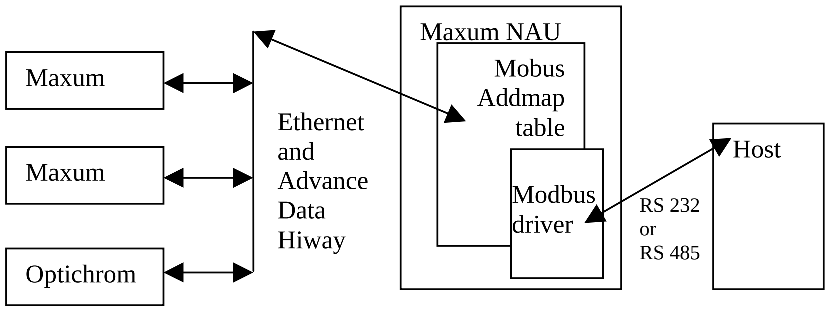
The following steps are illustrated in Figure 1.

- The Maxum units send sql commands or Optichrom Advance 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 Driver.

From the Host

- The Host sends a message to the MODBUS Driver (messages must be alternating in version 3.0 and 3.1)
- The MODBUS Driver processes a read request and returns a message to the host or sends the value to the modbus_addmap table.
- The MODBUS_addmap table determines the correct message to send to a specific Maxum or Optichrom Advance unit based on the address value_type, anlz, application, stream, and result.

Figure 1: Operation



System Specifications

Hardware Configuration

To configure hardware, have:

- Maxum and Optichrom Advance analyzers and an Advance Network Access Unit (NAU) attached to the same ethernet or Advance Data-NET network. Small systems can allow the Maxum GC to do its own MODBUS communications.
- NAU with 32-MB dynamic RAM and 8-MB battery backed up RAM.
- Host computer, setup to be a MODBUS master using MODBUS RTU messages, connected to the NAU using the NAU's RS-232 or RS-485 serial ports.

Serial Communication

To set up serial communication:

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

Y:a,b,c,d where

Y = serial port address (1 for the RS-232, 2 for RS-485) for SAM use "COM 1"

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

b = parity (e, o or n)

c = data bits (8 only)

d = stop bits

default: 1:19200,n,8,1

These settings can be changed at any time prior to or during MODBUS communication.

Note: For version 3.0 and 3.1, when configuring the host to send MODBUS messages, messages must be alternated, i.e., the NAU will discard "duplicate" messages unless a different message intervenes. This is no longer required for 3.11.

Unit/Stream/ Component Limits

There are no limits on the number of analyzers/units. Streams must have stream number >=100. Results must have result # >500. See the Address Map limits discussed on page 16. There are no requirements for analyzer 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 Maxums.

Component Values

Result Values

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

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

Scaled Results

Scaled results are stored in a single register as:

scaled result = Round(ScaleFactor * Maxum ResultValue / EUHI)

EUHI must be >= 1.0.

Floating Point

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).

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

Data Validity and Operational Checks

Status Information

Each Maxum or Optichrom Advance 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 forced always stream
- 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

Maxum Alarms

Maxum alarms are defined as any alarms that may occur on the Maxum analyzers. 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.

NAU Alarms

NAU alarms are defined as any alarms that may occur on the Advance Network Access Unit (NAU):

Alarm	Meaning
699	The Maxum tried to transmit a result or analyzer status that was not defined in the address map.
700	The Maxum analyzer is not reachable or refuses connection. It will not receive a command from the HOST.
701	Scale factor or EUHI is missing.

Alarm	**Meaning**
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 ad-dress map.
706	Can’t locate EUHI.
708	The Data_type does not match the value_type. Correction has been taken.

Host / Analyzer Communications

Description

This table summarizes the information types and availability. Topics in the following pages provide details for each information type.

Availability	Information
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 ad-dress map	Readme flags (Value_type: rdme)
	Database change flags (Value_type: DCHG)
	EUHI change flags (Value_type: ECHG)
	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)
	Alarm(Value_type: ALARM)
Host Controls sent to Maxum 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, SCYEAR)
	Clear Alarms (Value_type:CLEARALARM)
	Program Run (Value_type: PROGRAMRUN)
	DO set (Value_type: DOSET)

Readable from Host Computer

**Result values (Value_type: 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 Maxum analyzer from the Maxum's 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.

EUHI values (Value_type: EUHI) Valid Addresses: 30001-49999

This is the full scale value for each scaled result. It is set for each transmitted result in the Maxum'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 Maxum. EUHI values must be >= 1.0.

**Analyzer/Application status (Value_type: ANALYZERSTATUS)
Valid Addresses: 30001-49999**

Each stream transmits the application status at the end of cycle. The status values can range from 0 to 1000, as follows:

1000	normal, error free operation
9ss	Optichrom Advance warning alarm on stream ss
7ss	auto cal 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)

Information Available When Defined in Address Map

Readme flags (Value-TYPE: RDME) Valid Addresses: 0001-19999

This status flag can be defined for each analyzer/application/stream. It is set when the Maxum 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….). When the host reads a flag in any certain block, all the flags in that block are reset. The customer must be sure to read all rdme flags in a block in a single message. Starting with version 3.11, 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.

DIREAD flags (Value-TYPE: DIREAD) Valid Addresses: 0001-19999

This flag 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.

DOREAD flags (Value-TYPE: DOREAD) Valid Addresses: 0001-19999

This flag 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. The value in the result field designates the ID from the APPDO table.

Database change flags (Value_type: DCHG)
Valid Addresses: 0001-9999

This flag 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.

EUHI change flags (Value_type: ECHG)
Valid Addresses: 0001-9999

This flag can be defined for each analyzer/application/stream and is set when the EUHI is changed from the Maxum analyzer or from the Host. The flag is reset, after 3 seconds, when read by the Host. 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.

Information Available When Defined in Address Map, continued

Stream active flags (Value_type: CURRENTSTREAM)
Valid Addresses: 0001-9999

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

Stream skip flags (Value_type: SKIPSTREAM)
Valid Addresses: 0001-9999

This flag can be defined for each analyzer/application/stream. It is set by the Maxum 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 Maxum analyzer to disable or enable the stream.

Dedicated Stream (Value_type: DEDICATEDSTREAM)
Valid Addresses: 30001-49999

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

Alarm status (Value_type: ALARM)
Valid Addresses: 30001-49999

This integer (data_type I) can be defined for each analyzer, analyzer/application, and analyzer/application/stream. It contains the latest alarm code for fault alarms. Note that the ALARM defined for the analyzer corresponds directly with the red light on the front of the MMI. An ALARM defined for the stream or application, will reflect the contents of the curr_error attributes 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.

Analyzer standby (Value_type: STANDBY)
Valid Addresses: 0001-9999

This flag can be defined for each analyzer/application and is set by the Maxum analyzer. The flag tells the host if the application is in hold, out of service, or disabled. If set by the Host, a message is sent to the Maxum analyzer to place the application in hold (1 or True) or run (0 or False).

Information Available When Defined in Address Map, continued

Calibration (Value_type: CALIBRATE)
Valid Addresses: 0001-9999

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

Cycle Length (Value_type: CYCLELENGTH)
Valid Addresses: 30001-49999

This integer (data_type S or I) is set by the Maxum analyzer to be the longest cycle time in the Maxum's method table for an application, even unused methods.

Host Computer Settings

The Host can direct an application to start or stop running – see page 12, Analyzer Standby.

Analyzer control (Value_type: STANDBY)
Valid Addresses: 0001-9999

Calibration control (Value_type: CALIBRATE)
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 Maxum 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, it will be placed in run state. 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 Maxum MODBUS interface is implemented for Advance Optichrom analyzers.

Stream control (Value_type: SELECTSTREAM. SKIPSTREAM)
Valid Addresses: 0001-9999

The selectstream flag, 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 Maxum stream. The skipstream flag can be set to control Maxum stream enable/disable, as described in the previous section.

Host Computer Settings, continued

EUHI change (Value_type: EUHI) 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, page 29. Version 3.0 requires that EUHI values to be greater that 1.0.

Set clock (Value_types: SCMIN, SCSEC, SCHR, SCDAY, SCMON, SCYEAR) Valid Addresses: 40001-49999

By writing to the appropriate locations, the Host can set the date and time of day on the Advance Network Access Unit (NAU).

If the NAU has designated another system as its time server, this 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 Maxum analyzers. In order for the date and time to be sent to the Maxum analyzers, each Maxum analyzer will have to have its time server set to the NAU's IP address.

DOset flags (Value_type: DOSET)
Valid Addresses: 0001-9999

This flag can be defined for each analyzer/application/appdo and is set by the host.

Alarm Clear flags (Value_type: CLEARALARM)
Valid Addresses: 0001-9999

This flag can be defined for each analyzer, analyzer/application, and analyzer/application/stream. The Maxum 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 Maxum. If stream or application alarms are cleared, record of them is still kept by the application and stream until a cycle without alarms completes.

Program run (Value_type: PROGRAMRUN)
Valid Addresses: 40001-49999

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

Maxum MODBUS Address Map

Description

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

- Maxum analyzer results to be placed at certain addresses so the host computer can read them.
- control requests sent from the host computer to be directed to Maxum 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 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” 3 seconds after the host reads it. The value column in the Modbus_addmap table in System Manager 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 Advance System Manager/Tools/Advance Tools/Loader/MODBUS Load. An unload utility is also available.

General 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.

New in 4.0: As the map is loaded onto the analyzer using Advance 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.

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.

Creating and Loading a Map

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

Load the map onto the NAU with System Manager.

Use the following to configure Maxums 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 autotrt attribute in the stream_method table.
3. Designate NAU to receive results, using the host table

   type:6, anlzref:NAU's anlz_id

   or

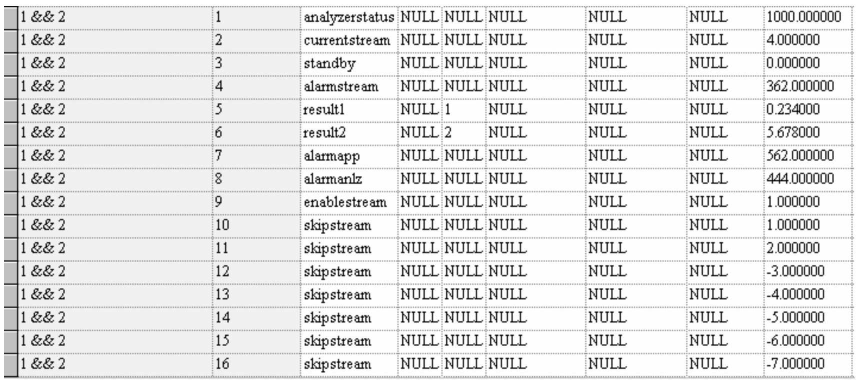
   Version 3.1 (after patch): The following format was developed for use by NAUs that are used to gather results from 3rd party analyzers and transmit them to a host. This will allow for running a MaxBasic program that transmits by frequency using the trtnow attribute in the stream_method table. An application 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 to application alarm
   Dedicatedstream – set to dedicated stream
   Cyclelength – set to cycle length
   Skipstream – set saved value to stream that is enabled (positive) or disablednegative)
   Calibrate – set to 1 if in calibration

Maxum MODBUS Address Map, Continued

Creating and Loading a Map, continued

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 Maxum 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	alarmanlz	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

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

Address 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. Note: ignored by the NAU.
Host_Tag Name	Character		Arbitrary, for use in referencing host database. This is the label assigned by the host to the data item. Note: ignored by NAU.
Description	Character		If blank, will be set by the load routine. As results transmit, this will contain the result_ name from the result table.

Address Entries, continued

Field Name	Type	Range	Description
Data_Type	Charac-ter		**S** for scaled fraction (only posi-tive 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
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 are re-quired 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 mem-ory locations. These address ranges allow the host to format the date 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.

Address Entries, Continued

Field Name	Type	Range	Description
Value_type	Character		(only the first 2 characters are required – case insensitive) **HO**STALIVE = host alive flag **RE**SULT=analyzer result value **CY**CLELENGTH=length of cycle **EU**HI=Euhi **RD**ME = readme flag **SE**LECTSTREAM = stream force flag **SK**IPSTREAM = stream skip flag **DE**DICATEDSTREAM = dedi-cated stream **CU**RRENTSTREAM = current stream flag **AN**ALYZERSTATUS= analyzer status **DC**HG= database change flag **EC**HG= euhi change flag **ST**ANDBY = standby flag **CA**LIBRATE= calibration flag **PR**OGRAMRUN = execute event **DO**SET = set a DO **DI**READ = read a DI on the NAU **AL**ARM = fault alarm code **CL**EARALARM = clear Maxum alams Clock settings: **SCSEC; SCMIN, SCHR, SCDAY, SCMON, SCDOW**
Initvalue	Character		Initial value (not for results or euhi)
Slave_ address	Integer		MODBUS Slave address
Slave_ address2	Integer		Second slave address (for re-dundancy) for future use
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, cy-cle_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.

Maxum MODBUS Address Map, Continued

Sample Address Map Lines

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

Ref_index, host, tag, desc, data_type, address, anlz, 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, ,
 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, ,
```

Maxum MODBUS Address Map, Continued

View/Edit Address Map

Once the address map has been loaded onto the NAU, it may be viewed and edited with System Manager/System Tables/MODBUS_addmap or MMI/Menu/Setup/Other Devices/MODBUS. Starting with Version 3.1, a MODBUS Address Map may be fully edited in place. Use System Manger to add, delete, and modify table entries in the Modbus_addmap or Modbus_addmap_result tables. Similar functions are available through the Maxum MMI. Since the MODBUS address is the primary key of the table, it cannot be modified, only deleted and replaced with a new entry.

The value attribute can be changed for testing. It contains the pre-scaled version of scaled results and character representations of Boolean values (True or False).

Address Map and MODBUS Driver

The MODBUS_addmap table acts as data collector for the Maxum and Optichrom analyzers and the NAU's MODBUS driver. The MODBUS driver has its own tables and memory. MODBUS communication occurs between the driver tables and the host computer. These tables are not viewable.

Example Map Configurations

Simple Example

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 Maxum units that is needed by the Host is results and analyzer status.

These addresses do *not* conform with the HCI-H.

Simple Example Map

	host tag	desc	data_ type	address	anlz	app	stream	res	value_ type	init value	slave 1	slave 2	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.

Detailed Example

Analyzer 162 has 2 applications:

- Application 1 has stream 1 with results 1-5
- Application 2 has stream 1 with results 1-5

All possible controls and information from the Maxum are available.

The addresses in this example *do* conform with the HCI-H.

Detailed Example Map

indx	host tag	desc	data_ type	ad- dress	anlz	app	stream	res	value_ type	init value	slave 1	slave 2	Euhi ref
1									sf	9999	1		
2									bv	65535	1		
2			B	13	162	1		1	DOSET	0	1		
2			B	14	162	2		1	DOSET	0	1		
2			B	15	162	2		2	DOSET	0	1		
3			B	11	162	1	1		CLEAR	0	1		
3			B	12	162	1	2		CLEAR	0	1		
3			B	151	162	1	0		ECHG	0	1		
4			B	152	162	2	0		ECHG	0	1		
5			B	201	162	1	0		DCHG	0	1		
6			B	202	162	2	0		DCHG	0	1		
7			B	351	162	1	0		CAL	0	1		
8			B	352	162	2	0		CAL	0	1		
9			B	1001	162	1	1		SKIP	0	1		
10			B	1002	162	2	1		SKIP	0	1		
11			B	1051	162	1	1		SEL	0	1		
12			B	1052	162	2	1		SEL	0	1		
13			B	1101	162	1	1		CUR	0	1		
14			B	1102	162	2	1		CUR	0	1		
15			B	1151	162	1	1		ECHG	0	1		
16			B	1152	162	2	1		ECHG	0	1		
17			B	1201	162	1	1		DCHG	0	1		
18			B	1202	162	2	1		DCHG	0	1		
19			B	1351	162	1	1		CAL	0	1		
20			B	1352	162	2	1		CAL	0	1		
21			B	1667	162	1			STBY	0	1		
22			B	1668	162	2			STBY	0	1		
23			B	2001	162	1	2		SKIP	0	1		
24			B	2002	162	2	2		SKIP	0	1		
25			B	2051	162	1	2		SEL	0	1		
26			B	2052	162	2	2		SEL	0	1		
27			B	2101	162	1	2		CUR	0	1		
28			B	2102	162	2	2		CUR	0	1		
29			B	2151	162	1	2		ECHG	0	1		
30			B	2152	162	2	2		ECHG	0	1		
31			B	2201	162	1	2		DCHG	0	1		
32			B	2202	162	2	2		DCHG	0	1		
33			B	2351	162	1	2		CAL	0	1		
34			B	2352	162	2	2		CAL	0	1		
35			B	3001	162	1	3		SKIP	0	1		
36			B	3002	162	2	3		SKIP	0	1		
37			B	3051	162	1	3		SEL	0	1		
38			B	3052	162	2	3		SEL	0	1		
39			B	3101	162	1	3		CUR	0	1		
40			B	3102	162	2	3		CUR	0	1		
41			B	3151	162	1	3		ECHG	0	1		
42			B	3152	162	2	3		ECHG	0	1		
43			B	3201	162	1	3		DCHG	0	1		
44			B	3202	162	2	3		DCHG	0	1		
45			B	3351	162	1	3		CAL	0	1		
46			B	3352	162	2	3		CAL	0	1		
47			B	4001	162	1	4		SKIP	0	1		
48			B	4002	162	2	4		SKIP	0	1		
49			B	4051	162	1	4		SEL	0	1		
50			B	4052	162	2	4		SEL	0	1		

Example Map Configurations, Continued

Detailed Example Map, continued

indx	host tag	desc	data_ type	address	anlz	app	stream	res	value_ type	init value	slave 1	slave 2	Euhi ref
51			B	4101	162	1	4		CUR	0	1		
52			B	4102	162	2	4		CUR	0	1		
53			B	4151	162	1	4		ECHG	0	1		
54			B	4152	162	2	4		ECHG	0	1		
55			B	4201	162	1	4		DCHG	0	1		
56			B	4202	162	2	4		DCHG	0	1		
57			B	4351	162	1	4		CAL	0	1		
58			B	4352	162	2	4		CAL	0	1		
59			B	2667	162	1			ECHG	0	1		
60			B	2668	162	2			ECHG	0	1		
61			B	3667	162	1			DCHG	0	1		
62			B	3668	162	2			DCHG	0	1		
63			B	6667	162	1			CAL	0	1		
64			B	6668	162	2			CAL	0	1		
76			B	10003	141	999		1	DIREAD	0	1		
76			B	10004	141	999		2	DIREAD	0	1		
65			B	11001	162	1	1		RDME	0	1		
66			B	11002	162	2	1		RDME	0	1		
67			B	12001	162	1	2		RDME	0	1		
68			B	12002	162	2	2		RDME	0	1		
69			B	13001	162	1	3		RDME	0	1		
70			B	13002	162	2	3		RDME	0	1		
71			B	14001	162	1	4		RDME	0	1		
72			B	14002	162	2	4		RDME	0	1		
73			B	15001	162	1	5		RDME	0	1		
74			B	15002	162	2	5		RDME	0	1		
75			S	30001	162	1			AN	0	1		
76			S	30002	162	2			AN	0	1		
77			I	30253	162	1	1		ALARM	0	1		
77			I	30254	162	1	2		ALARM	0	1		
77			S	30255	162	1	2		CY	0	1		
78			S	30256	162	2			CY	0	1		
79			S	30509	162	1			DE	0	1		
80			S	30510	162	2			DE	0	1		
81			I	40033					SCSEC	0	1		
82			I	40034					SCMIN	0	1		
83			I	40035					SCHR	0	1		
84			I	40036					SCDAY	0	1		
85			I	40037					SCMON	0	1		
86			I	40038					SCYR	0	1		
87			I	40039					SCDOW	0	1		
77			I	40240	162	1	1		PROG	0	1		
77			I	40241	162	1	2		PROG	0	1		
88			S	41101	162	1	1	1	RES		1		41151
89			S	41102	162	2	1	1	RES		1		41152
90			E	41151	162	1	1	1	EUHI		1		
91			E	41152	162	2	1	1	EUHI		1		
92			S	41201	162	1	1	2	RES		1		41251
93			S	41202	162	2	1	2	RES		1		41252
94			E	41251	162	1	1	2	EUHI		1		
95			E	41252	162	2	1	2	EUHI		1		
96			S	41301	162	1	1	3	RES		1		41351
97			S	41302	162	2	1	3	RES		1		41352
98			E	41351	162	1	1	3	EUHI		1		
99			E	41352	162	2	1	3	EUHI		1		

Detailed Example Map, continued

indx	host tag	desc	data_ type	address	anlz	app	stream	res	value_ type	init value	slave 1	slave 2	Euhi ref
100			S	41401	162	1	1	4	RES		1		41451
101			S	41402	162	2	1	4	RES		1		41452
102			E	41451	162	1	1	4	EUHI		1		
103			E	41452	162	2	1	4	EUHI		1		
104			S	41501	162	1	1	5	RES		1		41551
105			S	41502	162	2	1	5	RES		1		41552
106			E	41551	162	1	1	5	EUHI		1		
107			E	41552	162	2	1	5	EUHI		1		
108			S	42101	162	1	2	1	RES		1		42151
109			S	42102	162	2	2	1	RES		1		42152
110			E	42151	162	1	2	1	EUHI		1		
111			E	42152	162	2	2	1	EUHI		1		
112			S	42201	162	1	2	2	RES		1		42251
113			S	42202	162	2	2	2	RES		1		42252
114			E	42251	162	1	2	2	EUHI		1		
115			E	42252	162	2	2	2	EUHI		1		
116			S	42301	162	1	2	3	RES		1		42351
117			S	42302	162	2	2	3	RES		1		42352
118			E	42351	162	1	2	3	EUHI		1		
119			E	42352	162	2	2	3	EUHI		1		
120			S	42401	162	1	2	4	RES		1		42451
121			S	42402	162	2	2	4	RES		1		42452
122			E	42451	162	1	2	4	EUHI		1		
123			E	42452	162	2	2	4	EUHI		1		
124			S	42501	162	1	2	5	RES		1		42551
125			S	42502	162	2	2	5	RES		1		42552
126			E	42551	162	1	2	5	EUHI		1		
127			E	42552	162	2	2	5	EUHI		1		

MODBUS Messages

Supported Functions

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 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)

See page 36 for a detailed description of the MODBUS RTU message format.

Floating Point Formats

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 Maxum 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 Maxum Modbus Address Map defines which Modbus registers hold which type of data. The Maxum 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 Maxum 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																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	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 Maxum 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).

EUHI 16 Bit format

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:

Scaled Result = Round (ScaleFactor * Result value/ 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 Maxum system.

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

ResultValue = EUHI * Scaled Result/ 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 non-zero 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 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):

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

S	EXPONENT						MANTISSA								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte 0 (More Significant)								*Byte 1 (Less Significant)*							
16-Bit Word															

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 & 0x1FF;

        temp =  man;
        output = temp / (float)(1 << (9 - exp));
        output += 1 << exp;
        return output;
  }
```

"Empty Database"

On an "empty database" do these things.

1. Configure MODBUS slave and serial port: Using Advance System Manager, enter a value in the modbus_setting attribute of the System_Control table.

2. If an address map does not exist, create one using Microsoft Excel®. You will need to know the analyzer #, application #, stream #, and trtval # for every result that the system needs to handle. You will also need to know the specific customer requirements for host computer controls and information. See the Formula for Map Size section to determine if the system can handle the number of addresses required. 4000 addresses are a good rule of thumb for this limitation, but certain configurations may allow more or less addresses. The reason for this is that a map with more flags defined, as opposed to results, takes up less room than a map with mostly results. The limit for a map with only analyzer status and results may be less than 4000. Note that a much larger map may be successfully loaded, but when the NAU goes through a reset, the database cannot be rebuilt.

3. Load the address map: Use Advance System Manager's Load MODBUS utility (under Tools/Advance Tools/Loader/Load MODBUS) to add the address map to the NAU.

Set Up Maxum Analyzers

1. Use the Advance System Manager to add the NAU to the host table of the "sending" analyzers. Use Type 6 and set anlzref 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 Advance System Manager 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).

Set Up Optichrom Advance Analyzers

1. Set up the Optichrom Advance's results and events 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 under Advance System Manager/System Tables/System_control or on the MMI/Menu/Setup/System.

Special Instructions

- 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.
- Messages from the host must be alternated until version 3.11.
- 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.
- Starting with Version 3.1, it is acceptable to request a range of addresses, some of which do not exist. Bad values will be returned for the undefined addresses. The first address in the range must exist in the address map.
- Starting with Version 3.1, RS-485 multidrop is operational.

Formula for Map Size

Descriptions

1 x # of applications for application status
2 x # of results for results
1 for host alive (useless)
1 for database change flag
1 x # of applications for looking at longest cycle_length
1 x # of streams for readme flags
1 x # of streams for forcing streams
1 x # of streams for enable/disable stream
1 x # applications for determining if a stream is running dedicated
1 x # streams for determining next stream
1x # streams for euhi change flags + 1 x # applications
2 x # applications for calibration control
1 x # applications for run/hold

Example 1

Customer has 30 applications, 50 streams, 300 results

Customer wants no host control, merely unscaled results.

```
Map needs :     30 for application status
               300 for results
               330 addresses
```

Example 2

Customer has 150 applications, 400 streams, 600 scaled results

Customer wants readme flags, enable/disable stream, calibration control, application run/hold.

```
Map needs:  150 for application status
           1200 for results/euhi
            400 for readme flags
            400 for stream enable, disable
            300 for calibration control
            150 for run/hold
           2600 addresses
```

Formula for Map Size, Continued

Example 3

Customer wants 18 applications, 162 streams (9 per app), 1458 scaled results (9 per stream)

Customer wants all addresses as in HCI-H

Map needs:

```
   18 for application status
 2916 for results/euhi
  162 for readme
  162 for stream enable/disable
  162 for force stream
   18 for dedicated stream
   18 for run/hold
   36 for calibration
  180 for euhi change
  162 for next stream
   18 for cycle_length
__162 for select stream from host
 3852 addresses
```

MODBUS Primer

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 Maxum MODBUS implementation does not support broadcast messages.

Message Content

The message content is always:

Slave#	Function-Code	String_of_Data_Bytes
1 byte	1 byte	N bytes

Message Content

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 Maxum 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 (Function-Code). 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.

Protocol Formats

In the MODBUS protocol there are two formats for sending messages, MODBUS RTU and MODBUS ASCII (not supported by the Maxum). 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).

RTU vs. ASCII

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.

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.

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 3rd byte
2	6	3 + value of 3rd byte
3	6	3 + value of 3rd byte
4	6	3 + value of 3rd byte
5	6	6
6	6	6
7	2	3
8	6	6
9	2 + value of 5th byte	2 + value of 5th byte
10	2	2 + value of 5th byte
11	2	3 + value of 3rd byte
12	2	3 + value of 3rd byte
13	3 + value of 3rd byte	3 + value of 3rd byte
14	2	3 + value of 3rd byte
15	7 + value of 7th byte	6
16	7 + value of 7th byte	6
17	2	3 + value of 3rd byte
18	3 + value of 3rd byte	3 + value of 3rd byte
19	3 + value of 3rd byte	3 + value of 3rd byte
20	2 + value of 3rd byte	2 + value of 3rd byte
21	2 + value of 3rd byte	2 + value of 3rd byte
22-126	3 + value of 3rd byte	3 + value of 3rd byte
127	2	2
128-255	Communication Error	3

Message Length Table

Character Transmission

MODBUS protocol uses normal RS-232C, RS-422 (not supported in the Maxum), 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

Character Pause Relative to Baud Rate

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:
     (2.a)  Exclusive OR the message byte with the less significant
            byte of the checksum;
     (2.b)  Use the result of (2.a) as an index into the precomputed
            CRC-16 lookup table;
     (2.c)  Exclusive OR the result of (2.b) with the more significant
            byte of the checksum;
     (2.d)  Replace the current checksum with the result of (2.c);
```

CRC-16 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.e)   Place the result of (1.d) in the table at the index speci-
              fied by the value between 0 and 255.
```

Checksum and Communication Error

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.

MODBUS Data Types

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.

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, 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 be packed into the third byte which would have the value 0x05 (00000101) because the most significant 5 bits of the third byte would be set to zero:

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

Bit-Packing Digitals Into Bytes (bit positions are shown)

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.

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.

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.

MODBUS Functions

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)

Usually Supported Functions

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)

Very Common Functions

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)

Common Functions

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 Communications	check slave's communications counter

Less Common Functions

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

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

Function 01 – Read Coil Status

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

Read Coil Status Command

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

Read Coil Status Response

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.

Function 01 – Read Coil Status, Continued

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 on page 45, 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

ILLEGAL DATA ADDRESS Error Response

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

ILLEGAL DATA VALUE Error Response

Function 02 – Read Input Status

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

Read Input Status Command

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

Read Input Status Response

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.

Function 03 – Read Output Registers

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

Read Output Registers Command

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

Read Output Registers Response

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.

Function 04 – Read Input Registers

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

Read Input Registers Command

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

Read Input Registers Response

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.

Function 05 – Force 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

Force Single Coil Command

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

Force Single Coil Response

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

FAILURE IN ASSOCIATED DEVICE Error Response

Function 05 – Force Single Coil, Continued

Error Response

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.

Function 06 – Preset Single Register

Master Command

The master sends a command writing the value of one output analog (table 4), specifying the analog's address and the value to write. For example, the master might request that slave 17 write the value 926 to register 40136. Translating the values into hexadecimal, this is a request that slave 0x11 write 0x039E to the register at 0x0087 (corresponds to table offset 0136 of table 4):

Slave#	Function Code	Data Start		Analog Value	
		High	Low	High	Low
11	06	00	87	03	9E

Preset Single Register Command

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		Analog Value	
		High	Low	High	Low
11	06	00	87	03	9E

Preset Single Register Response

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).

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 134 (0x86) indicating that the error occurred on a command with Function Code 06.

Function 08 – Loopback Diagnostics Test

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)

Function 08 - Loopback Diagnotic Test Diagnostic Codes

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 on the next page.

Diagnostic Code 0

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

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

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 45) 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

Loopback Test Diagnostic Code 0 Command

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

Loopback Test Diagnostic Code 0 Response

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

Loopback Test Diagnostic Code 12 Command

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

Loopback Test Diagnostic Code 12 Response

Function 15 – Force 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

Force Multiple Coils Command

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

Force Multiple Coils Response

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 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.

Function 16 – Preset 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

Preset Multiple Registers Command

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

Preset Multiple Registers Response

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.

Maxum Implementation of MODBUS Protocol

Supported Functions

The Maxum will support the MODBUS RTU message format and a sub-set of MODBUS functions and error codes. The Maxum 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

Maxum MODBUS Function Codes

Exception Codes

The Maxum 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)

Maxum MODBUS Exception Response Codes

References

Siemens, Advance Maxum™ Gas Chromatograph (Version 1.0) User Database Table Descriptions, 8 September 1998

Siemens, Advance Maxum CD ROM Library, 2000583-001

Siemens, Advance Network Access Unit, User's Manual 2000581-001

Siemens, HOST Computer Communication Interface – HIWAY (HCI-H) for Advance Optichrom ® system, chapters 3 and 4, March 1993

ABB Hartmann & Braun Technical Note on MODBUS (draft) by Werner Ruediger, 24 April 1998.

Gould Inc. MODBUS Protocol Reference Guide PI-MBUS-300 rev. B, January 1985.

Schneider Group MODICON MODBUS Protocol Reference Guide PI-MBUS-300 rev D, chapters 1 and 2, appendices A and C, September 1997

Schneider Group Open MODBUS/TCP Specification (draft 2) by A. Swales, 3 September 1997

Preliminary Protocol Specification for the Maxum Modbus Protocol by R. D. Dillon, UCC, 22 November 1999

Index

address map
- bad value....19
- description....15
- edit or view....21
- entries....17
- examples....22
- how to load....16
- limits....16
- rules....15
- sample lines....20

alarms
- Maxum....7
- NAU....7

ASCII....36
bad value....19
character pause
- table....38

character transmission....38
checksum
- compute 16-bit CRC....39

codes
- diagnostic....54
- exception....60
- exception response....26
- Modbus....43
- supported functions....26

communication
- errors....36

configuration
- hardware....4
- host....4
- map....22

conversion routines
- EUHI....30

CRC-16
- lookup table....39

data types....41
- ranges....42

data validity....7
Definition of Terms....2
diagnostic codes....54
digital values
- message....41

DOREAD flags....11
edit
- map....21

Engineering Unit High....29
error
- checksum....40

error example
- function 08....56

error response
- function 01....46
- function 02....48
- function 03....49
- function 04....50
- function 05....52
- function 06....53
- function 15....58
- function 16....59

error responses
- function codes....44

errors
- communication....36

exception codes....60
floating point results....6
Force Multiple Coils
- function 15....58

Force Single Coil
- function 05....51

formula
- map size....33

Function 01....45
Function 02
- Read Input Status....47

Function 03
- Read Output Registers....49

Function 04
- Read Input Registers....50

Function 05
- Force Single Coil....51

Function 06
- Preset Single Register....53

Function 08
- error example....56
- Loopback Diagnostics Test....54

Function 15
- Force Multiple Coils....58

Function 16
- Preset Multiple Registers....59

function codes
- error responses....44
- Modbus....43

functions
- Modbus....43
- supported....26

getting started....31
- special instructions....32

HCI-H
- definition....29

implementation
- Modbus protocol....60

information
  status 7
  types 9
information types
  defined in map 11
limits
  address map 16
  component 5
  operational 5
  stream 5
  unit 5
Loopback Diagnostics Test
  function 08 54
map
  address 15
  bad value 19
  create or edit 16
  edit or view 21
  entries 17
  examples 22
  formula for size 33
  how to load 16
  limits 16
  rules 15
  sample lines 20
message
  digital values 41
  length 37
Modicon
  data types 42
Operation 3
operational checks 7
Preset Multiple Registers
  function 16 59
Preset Single Register
  function 06 53
protocol
  formats 35
  implementation 60
  Modbus 35
Read Coil Status
  Function 01 45
Read Input Registers
  function 04 50
Read Input Status
  Function 02 47
Read Output Registers
  function 03 49
references 61
response codes
  exception 26
results
  floating point 6
  scaled 6
ResultValue 29
RTU 36
  checksum 39
  restrictions 36
sample lines
  map 20
scale factor 19
scaled results 6
ScaleFactor 29
serial
  settings 4
set up
  analyzers 31
special instructions
  getting started 32
status information 7
values
  component 6
  result 6

Siemens Energy & Automation, Inc.
7101 Hollister Road, Houston, TX 77040
Phone 713-939-7400, Fax 713-939-9050

1/2007 Edition 2000584-001

Siemens Energy & Automation, Inc.

7101 Hollister Road, Houston, TX 77040
United States
Phone +1 (713) 939-7400
Fax +1 (713) 939-9050

www.usa.siemens.com/ia