GENESIS NETWORK

Genesis Controller DCS Communication Guide

THERMON

Genesis Controller DCS Communication Guide

Written and designed at Thermon, Inc.
100 Thermon Drive, San Marcos, TX 78667-0609, USA

PRODUCT WARRANTY INFORMATION

The seller warrants all equipment manufactured by it to be free from defects in workmanship or material under normal use and service. If any part of the equipment proves to be defective in workmanship or material and if such part is, within 12 months of the date of shipment from sellers factory, and if the same is found by the seller to be defective in workmanship or material, it will be replaced or repaired, free of charge, F.O.B. the seller's factory. The seller assumes no liability for the use or misuse by the buyer, his employees, or others. A defect within the meaning of this warranty in any part of any piece of equipment shall not, when such part is capable of being renewed, repaired, or replaced, operate to condemn such piece of equipment. This warranty is in lieu of all other warranties (including without limiting the generality of the foregoing warranties of merchantability and fitness for a particular purpose), guarantees, obligations, or liabilities expressed or implied by the seller or its representatives and by statue or rule of the law.

Contents

1. Genesis Controller Modbus Communication Protocols

The Genesis Controller heat trace controller is equipped with Ethernet ports, and can be readily connected to a distributed control system (DCS). The controller may be networked to a host PC, running windows-based Genesis Network client-server software, for remotely programming or monitoring and alarm status. The Genesis Controller supports the most widely used industrial Modbus protocol with standard Ethernet communication interface. When setting up the panel to communicate to a master device (likely a desktop computer), the unit is considered a slave device for Modbus. It is important that both the master device and the slave device are using the Modbus TCP/IP protocol (Figure 1). If the master device does not use the Modbus protocol, then a protocol conversion device must be used.

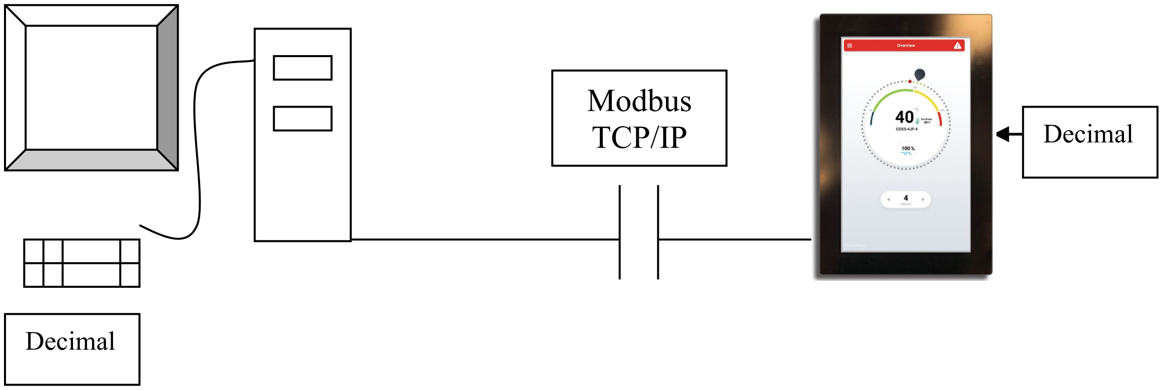

Figure 1. Conceptual block diagram of Modbus communication from master device to slave device with data displayed to user interface.

1.1 MODBUS TCP/IP Frame Format

The Modbus TCP/IP (or simple Modbus-TCP) is the Modbus RTU protocol with a TCP interface that runs on Ethernet. The Modbus-TCP message is simply a standard Modbus RTU data embedded into a TCP message (Figure 2). The message content of TCP/IP is equipped with checksum methods; hence, the Modbus checksum CRC is removed from the original the Modbus application PDU used in serial Modbus. The Modbus TCP transactions are functionally equivalent to serial counterparts with master and slaves exchange PDUs. Consequently, a Modbus TCP PDU includes the Modbus Application Protocol (MBAP) in addition to the traditional serial Modbus PDU. The MBAP header adds four fields, transaction identifier, protocol identifier, length and unit identifier.

- The transaction identifier allows devices to pair transaction requests and replies.
- The protocol identifier indicates the application protocol encapsulated by the MBAP header (zero for Modbus).
- The length field indicates the length in bytes of the remaining fields (unit identifier and PDU).

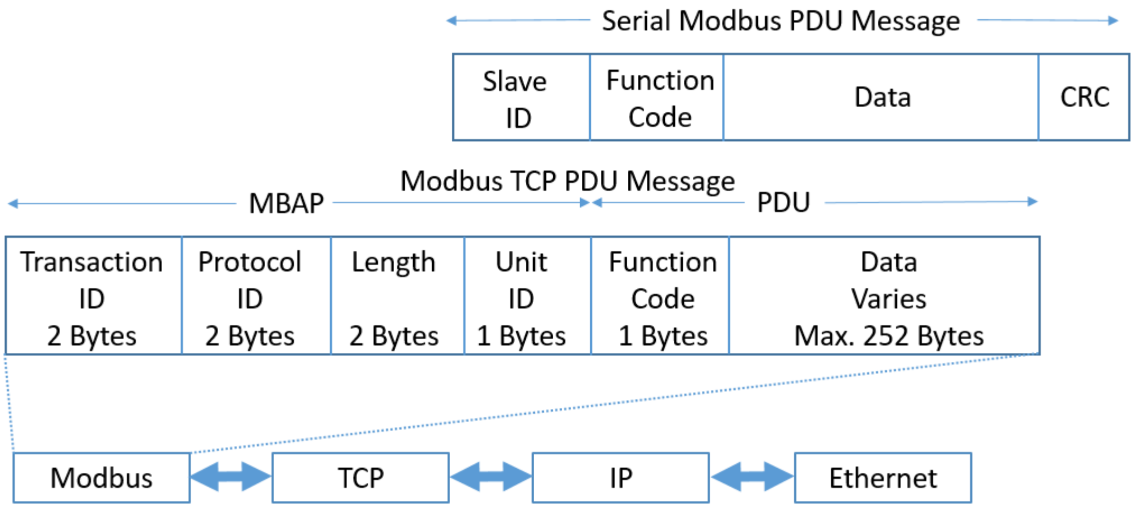

Figure 2. Modbus TCP Message

Name	Length (bytes)	Function
Transaction ID	2	For synchronization between messages of server and client
Protocol Identifier	2	0 for Modbus/TCP
Length Field	2	Number of remaining bytes in this frame
Unit Identifier	1	Slave Address (default value 0)
Function Code	1	Function codes as in other variants
Data Bytes	n	Data as response or commands

Table 1. Modbus TCP frame format

The Modbus RTU serial protocol's message data is in hexadecimal format (i.e. raw unconverted binary). The frame for the Modbus-TCP message is 12 bytes long for read requests, write requests, and write replies. Messages can be longer than 12 bytes, but not exceeding 260 bytes for read replies if more than one register is requested. Table 2 is an example of a read request, and Table 3 is an example of a write request.

Field	**Transaction ID**		**Protocol Identifier**		**Length Field**		**Unit Identifier**	**Function Code**	**Data Bytes**			
									Mem. Location		**No. Read**	
Byte#	1	2	3	4	5	6	7	8	9	10	11	12
Byte Value (hex)	0x00	0x0D	0x00	0x00	0x00	0x06	0x00	0x04	0x00	0x66	0x00	0x01
Byte Value (Decimal)	13		0		6		0	4	102		01	
Byte Value (Binary)	0000 0000	0000 1101	0000 0000	0000 0000	0000 0000	0000 0110	0000 0000	0000 0100	0000 0000	0110 0110	0000 0000	0000 0001

Table 2. Example Modbus TCP read request for Genesis Controller circuit #2 heater current

Field	Transaction ID		Protocol Identifier		Length Field		Unit Identifier	Function Code	Data Bytes			
									Mem. Location		Value	
Byte#	1	2	3	4	5	6	7	8	9	10	11	12
Byte Value (hex)	0x00	0x01	0x00	0x00	0x00	0x06	0x00	0x06	0x00	0xC9	0x01	0xC4
Byte Value (Decimal)	1		0		6		0	6	201		452	
Byte Value (Binary)	0000 0000	0000 0001	0000 0000	0000 0000	0000 0000	0000 0110	0000 0000	0000 0110	0000 0000	1100 1001	0000 0001	1100 0100

Table 3. Example Modbus TCP write request for Genesis Controller circuit #2 with maintain temperature set point value 452 which is equivalent of 45.2F.

1.2 Modbus Exception Codes

In a normal response, the slave repeats the function code. If an error occurs in the query received, the slave will return an exception message. In an exception response, the slave returns with the requested function code plus one byte of data, known as the exception code. Table 4 contains a list of exception code response used by the client/master application.

Code	Text	Details
1	Illegal Function	The function code received in the query is not allowed or invalid.
2	Illegal Data Address	The data address received in the query is not an allowable address for the slave or is invalid.
3	Illegal Data Value	Value is not accepted by slave
4	Slave Device Failure	Unrecoverable error occurred while slave was attempting to perform requested action
5	Acknowledge	Slave has accepted request and is processing it, but a long duration of time is required. This response is returned to prevent a timeout error from occurring in the master. Master can next issue a *Poll Program Complete* message to determine whether processing is completed
6	Slave Device Busy	Slave is engaged in processing a long-duration command. Master should retry later
7	Negative Acknowledge	Slave cannot perform the programming functions. Master should request diagnostic or error information from slave
8	Memory Parity Error	Slave detected a parity error in memory. Master can retry the request, but service may be required on the slave device
10	Gateway Path Unavailable	Specialized for Modbus gateways. Indicates a misconfigured gateway

11	Gateway Target Device Failed to Respond	Specialized for Modbus gateways. Sent when slave fails to respond

Table 4 Modbus Exceptions Code

2. Modbus Memory Map for the Genesis Controller

The Genesis Controller actively listens for incoming TCP connections on port 502 from master device connected on the same network. Information stored in Modbus memory map table can be accessed once a TCP communication channel has been established. The memory map is described in this section and referenced in the Global Settings and Circuit Settings tables.

Modbus read and write requests require a 16-bit data address to be referenced. The limits of the data address referenced in a Modbus message are from 0 to 9998 or 0x0000 to 0x270E. The data address range corresponds to either a read-only Function 04 Analog Input Registers 30001 to 39999, or the read/write Function 03/06 Analog Output Holding Registers 40001 to 49999.

The Genesis Controller only allows Modbus 04 reads and 03/06 read/writes. The Global Settings and Circuit Settings tables below describe the Genesis Controller memory map in more detail.

The Global Settings Modbus Data Addresses are referenced as listed in the “Base Memory Location” column of the Global Settings table. The Modbus Data Address for the non-global read-only Function 04 and read/write Function 03/06 table entries can be computed by adding the circuit number to the Base Memory Location.

Every Memory Location address as well as data length is two bytes or 16 bits.

Some DCS systems automatically put in an offset of plus or minus one for the memory location. As such, the memory locations should be checked against a known value to establish the automatic offset value.

Global Settings				
Function Code(s)	**Base Mem Location**		**Description**	**Allowed Values**
	Decimal	Hex		
4	10	0x00A	Logical Or of all alarm flags, All Circuits	Read Only

<table>
<tr><th colspan="5">Circuit Settings</th></tr>
<tr><th rowspan="2">Function Code(s)</th><th colspan="2">Base Mem Location</th><th rowspan="2">Description</th><th rowspan="2">Allowed Values</th></tr>
<tr><th>Decimal</th><th>Hex</th></tr>
<tr><td colspan="5">Add Circuit # x 100 to Base Memory Location to get the MODBUS Data Address
Circuit 1's   Heater Current = 2 + 1 *100 = 102 (0x0066)
Circuit 72's   Heater Current = 2 + 72*100 = 7202(0x1C22)
(Values below represent Circuit 1's location)</td></tr>
<tr><td>4</td><td>100</td><td>0x064</td><td>Control Temperature for the DCM
RTD Reading for the DTM</td><td>Value x10 (1)
Temperature Ranges</td></tr>
<tr><td>4</td><td>101</td><td>0x065</td><td>Which DTM and RTD that the circuit is controlling from</td><td>low Byte = DTM address 1-199
High Byte = RTD from DTM 1-6</td></tr>
<tr><td>4</td><td>102</td><td>0x066</td><td>Heater Current</td><td>Value x 10 (A) (2)</td></tr>
<tr><td>4</td><td>103</td><td>0x067</td><td>Ground Current</td><td>Value in (ma)</td></tr>
<tr><td>4</td><td>104</td><td>0x068</td><td>Heater Percent On</td><td>Range = 0-100%</td></tr>
<tr><td>4</td><td>105</td><td>0x069</td><td>Alarms</td><td>Bit definitions (3)
0x8000(bit) High Current Trip
0x4000(bit) Programing error
0x2000(bit) Current over .5A when circuit off
0x1000(bit High Ground Trip bit
0x0800(bit) RTD Fault No Communication for all assigned RTD's
0x0400(bit) High Temperature trip
0x0200(bit) RTD Fault (all assigned RTD's in fault)
0x0100(bit) Not Used
0x0080(bit) High Current
0x0040(bit) Low Current
0x0020(bit) Circuit Fault
0x0010(bit) High ground current
0x0008(bit) RTD Fault No Communication
0x0004(bit) High temperature Alarm
0x0002(bit) RTD Fault (one or more assigned RTD's in fault)
0x0001(bit) Low temperature Alarm</td></tr>
</table>

Circuit Settings				
Function Code(s)	**Base Mem Location**		**Description**	**Allowed Values**
	Decimal	Hex		
Add Circuit# x 100 to Base Memory Location to get the MODBUS Data Address Circuit 1's Maintain Temp = 1 + 1 *100 = 101 (0x0065) Circuit 72's Maintain Temp = 1 + 72*100 = 7201(0x1C21) (Values below represent Circuit 1's location)				
03/06	100	0x0064	Alarm Acknowledge	Bit Value [3] 0x8000(bit) High Current Trip 0x4000(bit) Programing error 0x2000(bit) Current over .5A when circuit off 0x1000(bit) High Ground Trip bit 0x0800(bit) RTD Fault No Communication for all assigned RTD's 0x0400(bit) High Temperature trip 0x0200(bit) RTD Fault (all assigned RTD's in fault) 0x0100(bit) Not Used 0x0080(bit) High Current 0x0040(bit) Low Current 0x0020(bit) Circuit Fault 0x0010(bit) High ground current 0x0008(bit) RTD Fault No Communication 0x0004(bit) High temperature Alarm 0x0002(bit) RTD Fault (one or more assigned RTD's in fault) 0x0001(bit) Low temperature Alarm
03/06	101	0x0065	Maintain temp	10x True Value
03/06	102	0x0066	Control Band	10x True Value
03/06	103	0x0067	High Temperature Trip Alarm	
03/06	104	0x0068	High Temp Alarm	
03/06	105	0x0069	Low Temp Alarm	10x True Value
03/06	106	0x006A	High Ground Fault Trip Alarm	Value in (ma)
03/06	107	0x006B	High Ground Fault Alarm	Value in (ma)
03/06	108	0x006C	High Current Alarm Trip	10x True Value (A)
03/06	109	0x006D	High Current Alarm	10x True Value (A)
03/06	110	0x006E	Low Current Alarm	10x True Value (A)

Circuit Settings

Function Code(s)	Base Mem Location		Description	Allowed Values
	Decimal	Hex		
Add Circuit# x 100 to Base Memory Location to get the MODBUS Data Address Circuit 1's Maintain Temp = 1 + 1 *100 = 101 (0x0065) Circuit 72's Maintain Temp = 1 + 72*100 = 7201(0x1C21) (Values below represent Circuit 1's location)				
03/06	111	0x006F	Circuit Enable/ Status	Bit definitions 0x0008(bit) Heater Forced Off = 1 normal = 0 0x0004(bit) Heater Forced On = 1 normal = 0 0x0002(bit) Heater Tripped = 1 normal = 0 0x0001(bit) Heater Enabled = 1 , Disabled = 0
03/06	112	0x0070	Control Type	0 = On/Off 1 = On/Off with a Soft Start 2 = Proportional 3 = Ambient Proportional Mechanical 4 = PID
03/06	113	0x0071	Number of RTDs per Circuit	1 to 20
03/06	114	0x0072	RTD Fault power	Range = 0 to 100%
03/06	115	0x0073	Power Clamp	Range = 0 to 100%

(1) **Temperature Ranges:**

Read: For °C, integer value read as -1289 to 6000 divided by 10 results in a temperature in the range of -128.9 to 600.0°C.
For °F, integer value read as -2000 to 11120 divided by 10 results in a temperature in the range of -200.0 to 1112.0°F.

Write: For °C, temperature in the range of -128.9 to 600.0°C multiplied by 10 results in integer value from -1289 to 6000 to be written.
For °F, temperature in the range of -200.0 to 1112.0°F multiplied by 10 results in integer value from -2000 to 11120 to be written.

RTD Open: If greater than 600.0°C or 1112.0°F

RTD Fault: If less than -128.9° C or -200.0°F

RTD Mapping: In circuits with multiple RTD mapping, the Control Temperature is selected for the RTD with the lowest temperature reading, or the highest temperature reading when any RTD reading exceeds the High alarm set point.

(2) **Current Ranges**

Read: Integer value read as 0 to 12500 divided by 100 results in a current in the range of 0.00 to 125.00 Amps.

Write: Current in the range of 0.00 to 125.00 Amps multiplied by 100 results in an Integer value from 0 to 12500 to be written.

(3) **Alarm/Trip/Fault Bit Settings**

Note: Circuit alarm may consist of multiple alarms.
For example: Circuit with low current and low temperature alarm will report alarm code of 41 (Hex) / 65 (decimal)

	Trip/Alarm/Fault Type	Binary / Hex Bit / Decimal
Upper	High Current Trip	1000 0000 / 0x8000 (bit) / 32,768
Byte:	Programing error	0100 0000 / 0x4000 (bit) / 16,384
	Current over .5A when circuit off	0010 0000 / 0x2000 (bit) / 8,192
	High Ground Trip bit	0001 0000 / 0x1000 (bit) / 4,096
	RTD Fault No Communication for RTD's	0000 1000 / 0x0800 (bit) / 2,048
	High Temperature trip	0000 0100 / 0x0400 (bit) / 1,024
	RTD Fault (all assigned RTD's in fault)	0000 0010 / 0x0200 (bit) / 512
	Not used	0000 0001 / 0x0100 (bit) / 256
Lower	High Current	1000 0000 / 0x0080 (bit) / 128
Byte:	Low Current	0100 0000 / 0x0040 (bit) / 64
	Circuit Fault	0010 0000 / 0x0020 (bit) / 32
	High ground current	0001 0000 / 0x0010 (bit) / 16
	RTD Fault No Communication	0000 1000 / 0x0008 (bit) / 8
	High temperature Alarm	0000 0100 / 0x0004 (bit) / 4
	RTD Fault (one or more assigned RTD's in fault)	0000 0010 / 0x0002 (bit) / 2
	Low temperature Alarm	0000 0001 / 0x0001 (bit) / 1

3. Help

Help Information

The DCS communications interface is intended to be used with the support of this instruction guide, the specific control module operating guide, and the specific DCS system operating manuals applicable. If special support needs do arise, Thermon provides local support through its area representatives and affiliate companies as well as through a toll-free user support line.

For support dial 1-800-820-HEAT.

Corporate Headquarters: 7171 Southwest Parkway • Building 300, Suite 200 • Austin, TX 78735 • Phone: 512-690-0600
For the Thermon office nearest you visit us at . . . www.thermon.com