



Modbus Communication Module



Technical Manual

(For Actuators with Software Above Rev.3.2)



ontracactuators.com

Table of Contents

1. Introduction.....	1
Caution 	2
2. General Information.....	3
Ontrac Modbus Module Installation	3
Wiring Diagram	4
3. Electrical and Technical Parameters.....	7
Operation Parameters.	7
4. Physical Layer Setup (RS-485 Connection).....	9
Network Topology	11
5. Modbus Description.....	13
The Request.....	13
The Response.....	14
Communication Mode (RTU Mode)	14
RTU Byte Format	15
RTU Framing.....	15
Address Field	16
Function Field.....	16
Data Field.....	17
Error Checking Field.....	17
Error Correction Mechanisms	18
Reference program in C language for CRC 16	18
6. Modbus Communication	20
Some Variable, Type, Description and Direction.....	20
Data Structure in Internal Memory	21
7. Actuator Control.....	29
Examples of Modbus Telegrams	29
Reading Data from the Actuator Holding Register	30
Error and Alarm Registers	30
Examples of Modbus Telegram for reading Errors.....	31
Examples of Modbus Telegram for reading Alarms	34
8. Device Configuration	36

Actuator and Modbus Configuration	36
9. Troubleshooting.....	37

1. Introduction

Ontrac offers a wide range of software and hardware products. The hardware products include Multitrac and the Easytrac family of actuators. Ontrac provides a wide range of options for controlling these actuators and one of them is the serial communication options through the use of an optional communications board called the Ontrac Modbus Module (OMM).

The OMM works with both Multitrac and Easytrac actuators. The (OMM) performs the tasks of network interface, it collects data from the actuator and it issues commands from the master controller to the actuator.



Figure 1 Family of Ontrac Actuators

The Ontrac Modbus Module (OMM) provides a serial connection to the actuator for data transfer. Through the OMM the actuator can receive commands, settings and configurations and can transmit a wide selection of status information. The exact commands are based on the actuator - Easytrac or Multitrac - and can be found in the Data Structure in Internal Memory section of this manual.

Ontrac offers Modbus RTU protocol communications at a baud rate up to 38.4kbits/s. The factory default speed is 9.6kbits/s giving a 1.5km maximum distance between nodes. This document describes the OMM that has been designed to fully comply with the Modbus protocol specification.

Caution

If any address or function that is not stated in this specification or other Ontrac documentation specification is sent to the actuator, the actuator may not operate properly or it may execute unintended movements. Please exercise due caution by not sending any function or address not specified by Ontrac.

Connect the Actuator to a computer running the Smartrac Software using the RS-485 cable, then set the communication speed and other parameters.

If the actuator is used or stored in a place or conditions that are outside the parameters specified by the manufacturer and without sufficient shielding measures, the actuator may malfunction if sufficient actions are not taken. The actuator should also not be allowed to operate in the following environment.

1. Where large current or high magnetic field is generated.
2. Where arc discharge occurs due to welding, etc.
3. Where electric noise generates due to electrostatic, etc.
4. Where the actuator may be exposed to radiation

2. General Information

The OMM implements a half-duplex (2-wire) RS-485 at the physical layer and it uses Modbus RTU protocol for the exchange of data and transmission of control signal over the data highway between actuators and a Modbus capable host system. There are other forms of Modbus; this includes the MODBUS TCP/IP.

MODBUS TCP/IP is a variant of the MODBUS family of simple, non-proprietary communication protocols used for supervision and control of automation equipment. The MODBUS TCP/IP covers the specifics of MODBUS messaging in an 'Intranet' or 'Internet' environment using the TCP/IP protocols stack. The most common use of the protocols at this time are for Ethernet attachment of PLC's, I/O modules, and 'gateways' to other simple field buses or I/O networks.

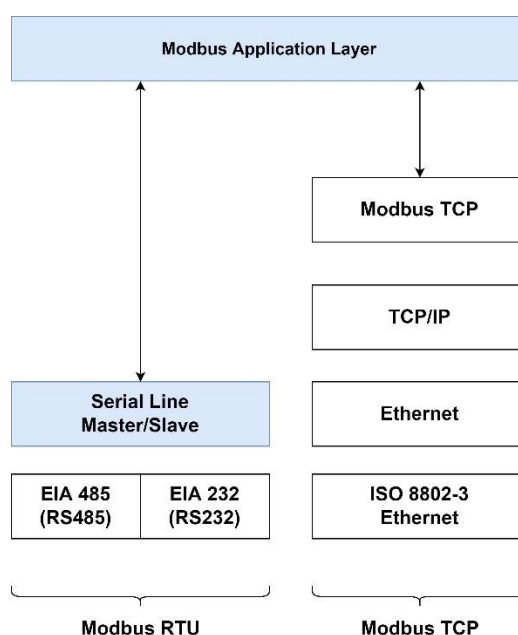


Figure 2 Modbus Stack Layers

Ontrac Modbus Module Installation

The Ontrac Modbus Module is integrated into the housing of the actuator and connected to the Central processing unit. The overall system diagram is shown in the diagram below. The Ontrac Modbus Module is shown as the field bus block in the diagram. The installation procedure varies with the type of actuator. A detailed guide for the installation of the module can be found in the manual of each Ontrac actuators.

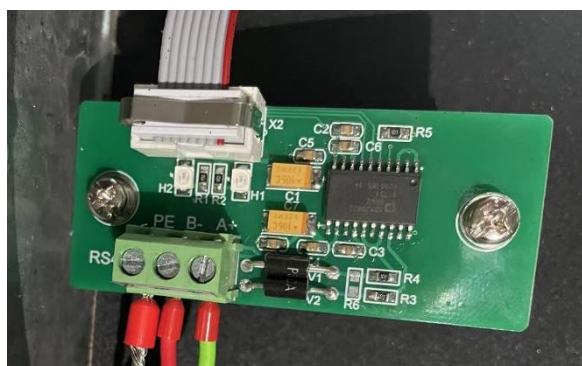


Figure 3 Easytrac Modbus Module



Figure 4 Modbus port for Multitrac

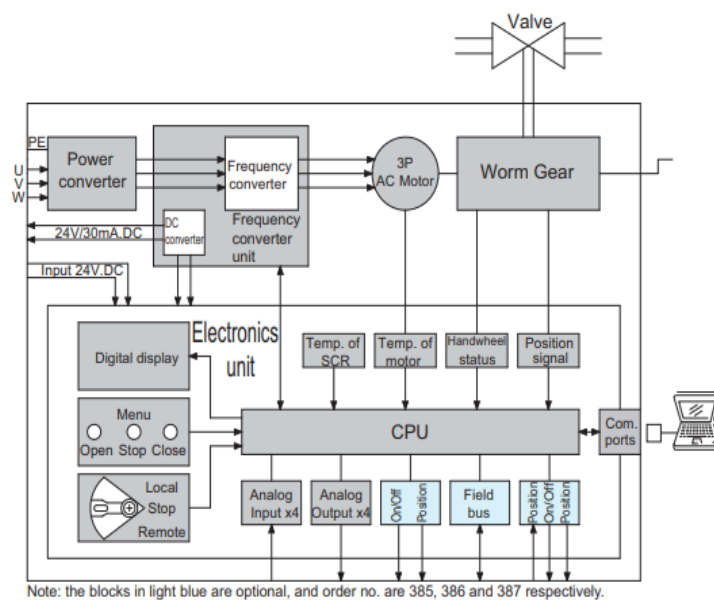


Figure 5 Actuator Block Diagram

Wiring Diagram

The Ontrac Cloud (**Cloudtrac**) gives users complete global access to an actuator's specific technical details and parameters without the use of special tools or proprietary software. Every Ontrac actuator comes with a permanent QR code attached. Simply scanning the QR (Quick Response) code on your actuator connects you to Cloudtrac. The cloud provides access to the specific information for the actuator.

Users will then be able to download instruction and data sheets as well as actuators' specific wiring diagram. This simple aid to set up, commissioning and fault finding will save users considerable time and ensure the right connections are made. An example of the actuator wiring diagram is shown in the diagrams below. In this case, the Data highway (RS 485) connects to terminals 23 and 24.

Electrical wiring for digital signal input:

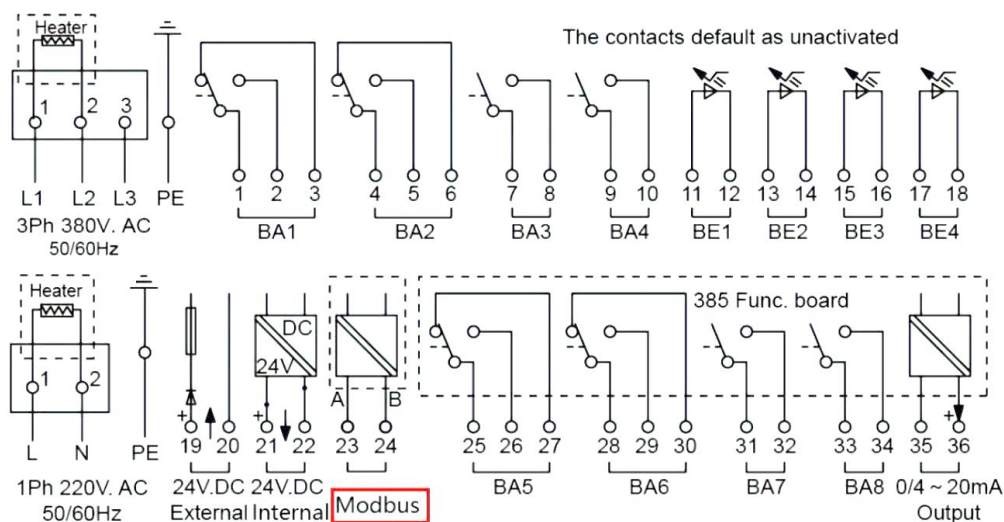


Figure 6 Electrical wiring for digital signal input of Multitrac

Electrical wiring for analog signal input:

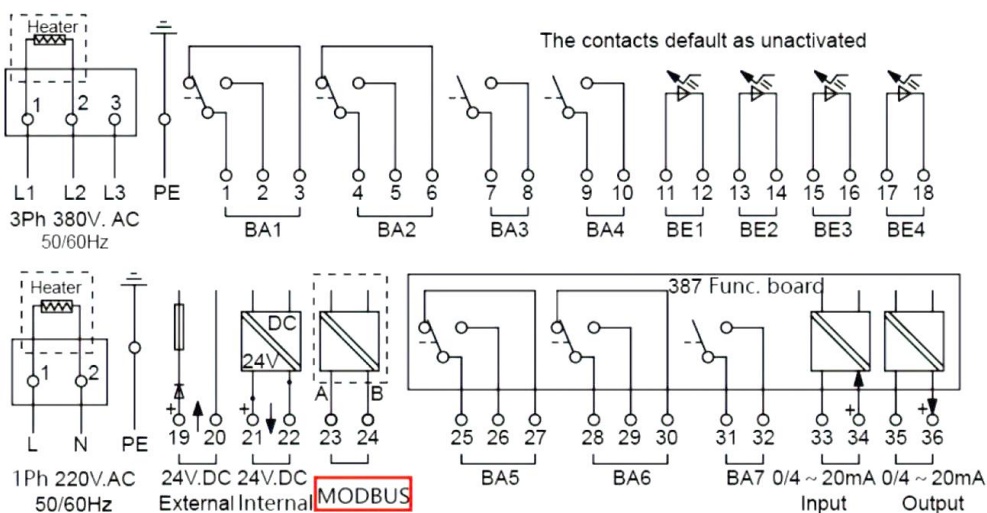


Figure 7 Electrical wiring for analog signal input of Multitrac

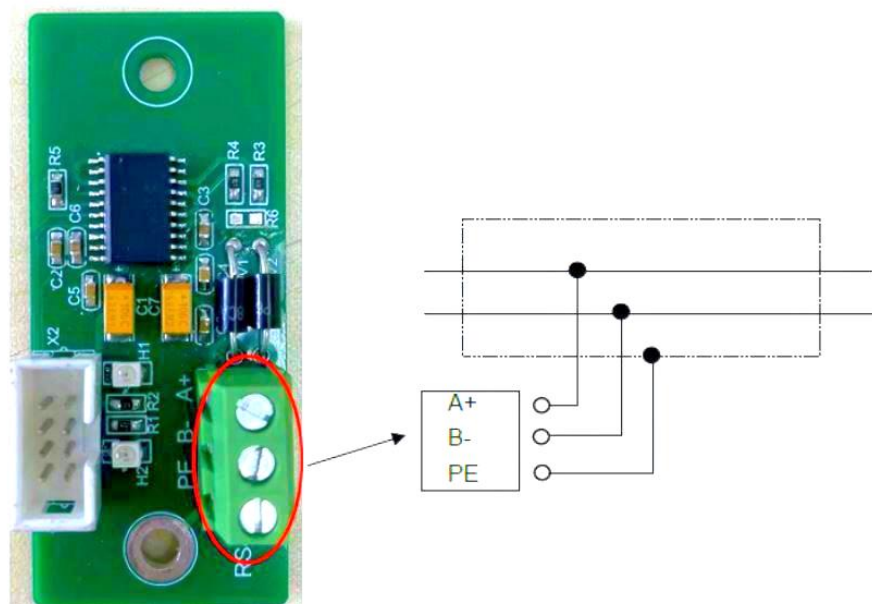


Figure 8 Electrical Wiring of Easytrac

3. Electrical and Technical Parameters

The circuit of the OMM is designed to communicate with other subsystems of the actuator through the internal communications bus. The actuator system is designed in such a way that the Ontrac Modbus Module (OMM) is not placed in the control path of the actuator and hence it does not interfere with the integrity and reliability of the actuator control.

The OMM has a dedicated microcontroller that runs the communication firmware, therefore the RS-485 data highway connections are fully isolated from the actuator electronics.

Operation Parameters.

Parameter	Value
Operating Voltage	12 - 24 V AC/DC $\pm$ 10%
Maximum Power Consumption	60.6 mW.
Communication Channel Interface	RS-485 (EIA 485)
Communication Protocol	Modbus RTU
Communication Parameters	Multitrac: Baud rate (bits per second): 9600. Data bits: 8 Parity bit: none Stop bits: 1
	Easytrac: Baud rate (bits per second): 1200, 2400, 4800, 9600, 14400, 19200. Data bits: 8 Parity bit: none Stop bits: 1
Topology	Multipoint
Transfer Type	Half Duplex

Max. Number of connected devices	32 (with repeaters larger, usually up to 256)			
Maximum Cable Length	1500 Meters			
Air Media	Without Corrosive flammable or explosive gas.			
Humidity	Humidity: 100% (<50°C) non-condensing.			
Recommended Storage Temperature	-20°C to +50 °C in dry environment.			
Operating Temperature	Multitrac:	Standard	Low Temperature	Very Low Temperature
	MOE On/Off:	-25°C to +70°C	-45°C to +70°C	-60°C to +70°C
	MME Modulating:	-25°C to +60°C	-40°C to +60°C	-60°C to +60°C
	Split Head:	-40°C to +85°C	-40°C to +85°C	-45°C to +85°C
	Easytrac:			
	-20°C to +65°C (Optional: -40°C to +60°C)			

4. Physical Layer Setup (RS-485 Connection)

RS-485 (EIA-485) is the most common interface in the industry because the RS-485 uses a multi-point or multi-drop topology. This topology allows the connection of several receivers and transmitter together. There are two types of RS-485; RS-485 with 2 contacts, this operates in half duplex mode. The other type is RS-485 with 4 contacts, this is capable of operating in the full duplex mode. The Ontrac Modbus Module (OMM) implements half duplex (2 Contacts).

A single segment of an RS-485 network can connect up to 32 devices, but this number can be expanded to 256 using additional repeaters and signal amplifiers. Only one device can transmit at a time, that is why the networks have one master and multiple slaves. The speed of operation also depends on the length of the cable. The shorter the cable the higher the transmission speed. The speed can reach 10 Mbit / s at 10 meters.

The RS-485 standard specifies the voltage on the lines to be in the range from -7 V to +12 V but the standard does not specify a type of connector. The pinout of the RS-485 connector depends on the manufacturer of the device and the purpose. For Ontrac actuators, there are three possible terminal options (Standard Terminals, Plug-In Terminals and Screw Terminals).

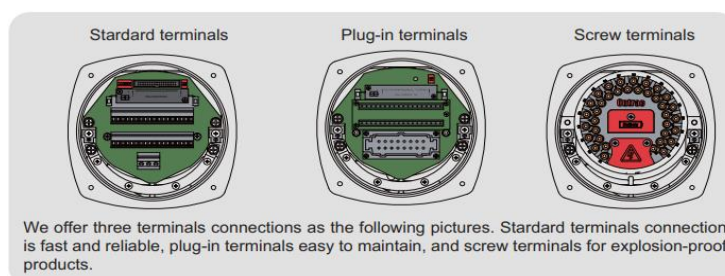


Figure 9 Terminals Supported by the Actuators (Multitrac)



Figure 10 Terminals Supported by the Actuators (Easytrac)

The Modbus terminal for a specific actuator can be obtained from the Cloudtrac. When the QR code on the actuator is scanned, the wiring diagram is and other technical information about it can be obtained.

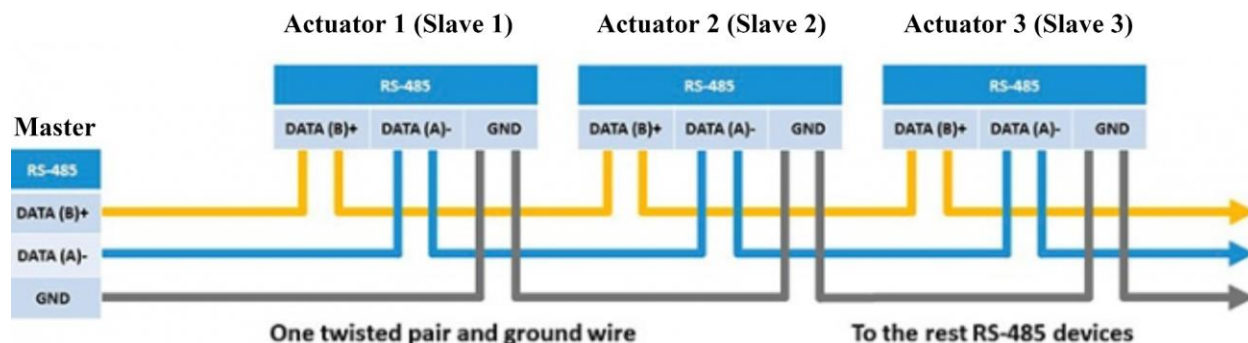


Figure 11 Wiring Scheme for half duplex Modbus RTU.

The RS-485 highway only supports signal transfer; it is not designed for power transfer. As a result of this, the actuator must provide the power to the OMM. This also means that the OMM cannot communicate with the Master device if power is not supplied to the actuator.

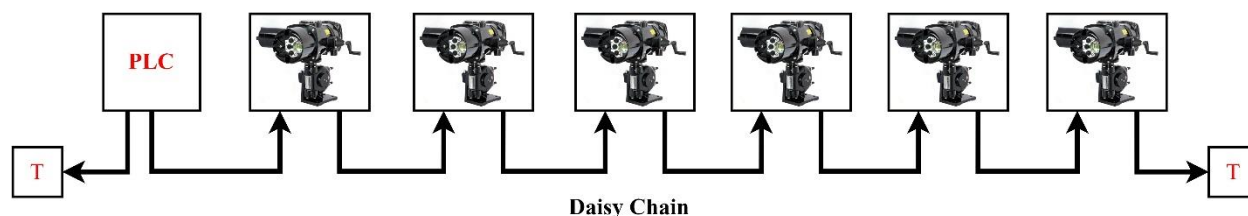


Figure 12 Modbus RTU Connection at the physical layer

The data highway must be terminated at both ends because of fast transitions, high data rates, and cable length. The purpose of the termination is to prevent adverse transmission-line phenomena, such as reflections. Termination resistors also reduce electrical noise sensitivity due to the lower impedance. The value of the termination resistors should be equal to the value of the characteristic impedance of the cable which is typically 120 ohms for a twisted pair cable.

One of the terminators must be an active 5VDC (Requires a 5V DC power supply) termination so that lines are biased to known voltages and nodes will not misinterpret noise from undriven lines as actual data. When a biasing network is not used, the undriven lines will be in an intermediate state which will result in high electrical noise sensitivity when all the connected devices are powered off or not transmitting data. An example of a biasing network is shown in the image below.

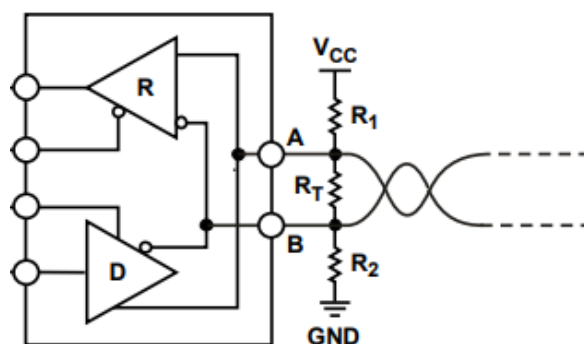


Figure 13 Active Termination

Network Topology

Two connection topologies are possible with the RS-485. The two topologies are bus topologies but the difference is in how connections are made. In the first topology, each device is connected to a single cable that is referred to as the backbone or the trunk line.

The devices are connected to the trunk line using spur lines or branch lines. This system requires that taps located in junction boxes are situated close to the devices. The second option is simply a daisy-chain connection between the devices. The two configurations require that the cumulative length of the lines (trunk line and spur lines) is kept to the minimum so that the needed communication speed is achievable.

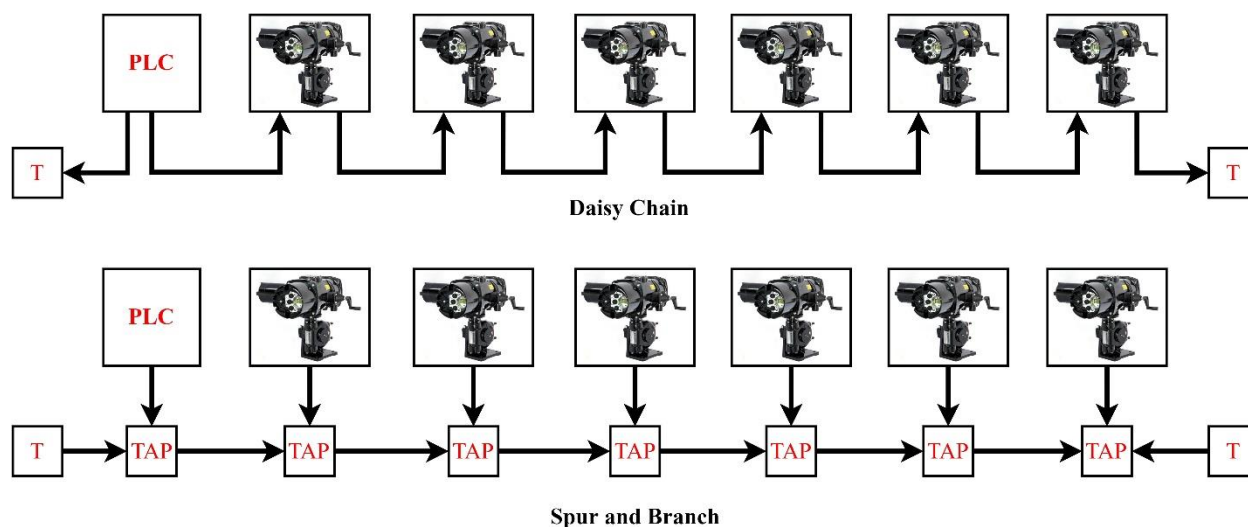


Figure 14 Modbus Connection Topologies

The table below shows the cumulative cable length at some standard transmission speed.

Data Rate (Baud)	9600	19200	38400	57600	115200
Max Cable Length	1.2 Km	1.2 Km	1 Km	0.75 Km	0.5 Km
Total Spur Length	0.5 m	0.5 m	0.3 Km	0.2 m	0.1 m

5. Modbus Description

The Controller communicates with devices that are connected to the same bus using query-response cycle monitoring at specific timer intervals. In this approach only one device (the master) can initiate transactions. This transaction is called a request. The other devices (the slaves) respond to requests from the master by supplying the requested data or by taking a specific action specified in the request.

Typically, a master device has inbuilt host processors and programming panels. On the other hand, most slaves have programmable microcontrollers and an ASIC (Application Specific Integrated Circuit) that serves as an interface with the physical layer. The master can identify and send a message to a specific slave by using its address and it can also initiate a broadcast request to all slaves.

Slaves respond to message that are addressed to them individually but they do not respond to broadcasts from the master. A MODBUS message sent from a master to a slave contains the address of the slave, the 'command' (e.g. 'read register' or 'write register'), the data, and a check sum (CRC). Modbus protocol is just a messaging structure; therefore, it is independent of the underlying physical layer. As mentioned in earlier sections, the OOM physical layer is implemented using RS485.

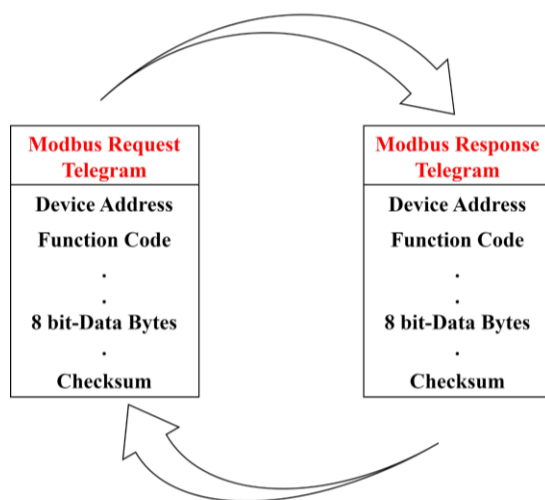


Figure 15 Dataflow between master and slaves in a Modbus Network

The Request

The function code is an instruction (a command) that tells the addressed slave device what kind of action to perform. The Ontrac Modbus Module uses four important function code, a description of these codes can be found in subsequent sections. The function code is inserted in the bytes of the request. Aside from the function code, other bytes also contain additional information that the slave will need to perform the function.

For Instance, function code 03h will request the slave to read holding registers and respond with their contents. The data field must contain the information about the start register and the number of registers to read. The error check field provides a mechanism that the slave can use to validate the integrity of received message.

The Response

When a package is sent to the slave, it must respond within a certain time period or the master will call it a "no response" error. If the slave makes a normal response, the function code in the response is the same as the function code in the request.

The data bytes contain the data collected by the slave. If the system encounters an error, the function code is changed to the function code of an error response, and the data bytes contain information about the nature of the error. The error check field stores a value that the master can use to confirm that the message contents are valid.

Communication Mode (RTU Mode)

The MODBUS protocol has two communication modes; The Modbus RTU, and Modbus ASCII. The Ontrac Modbus Module (OMM) is set up to communicate using the RTU mode. A complete Modbus request or response is referred to as a telegram or Protocol Data Unit (PDU). Each telegram consists of bytes and each byte is made up of bits.

The diagram below depicts the structure of a Modbus telegram. When controllers are set up to communicate on a Modbus network using RTU (Remote Terminal Unit) mode, each eight-bit byte in a message contains two four-bit hexadecimal characters. The main advantage of this mode is that its greater character density allows better data throughput than ASCII for the same baud rate. Each message must be transmitted in a continuous stream.

Content of Modbus Telegram
Device Address
Function Code
Register Number
Register Count
Data
Checksum

Figure 16 Content of Modbus Telegram or PDU

RTU Byte Format

Coding System	8-bit binary, hexadecimal 0–9, A–F Two hexadecimal characters contained in each 8-bit field of the message
Bits Per Byte	1 start bit 8 data bits, least significant bit sent first 1 bit for even/odd parity; no bit for no parity 1 stop bit if parity is used; 2 bits if no parity
Error Check Field	Cyclic Redundancy Check (CRC)

RTU Framing

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

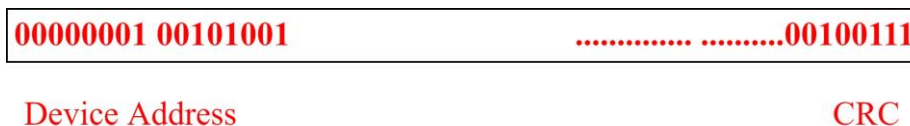


Figure 17 Modbus Frame

The allowable characters transmitted for all fields are hexadecimal 0–9, A–F. Networked devices monitor the network bus continuously, including during the ‘silent’ intervals. When the first field (the address field) is received, each device decodes it to find out if it is the addressed device.

Following the last transmitted character, a similar interval of at least 3.5 character times marks the end of the message. A new message can begin after this interval. The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame, the receiving device flushes the incomplete message and assumes that the next byte will be the address field of a new message.

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

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

The data in the actuator may be accessed or modified using different Modbus function codes. The commands sent by the master can be of two forms; register or discrete based. This flexible design allows any host device to control the actuator over a Modbus network in the simplest way. The following section provides more details about each section.

Address Field

The address field of a message frame contains eight bits. Valid device addresses are in the range of 01 – 126 decimal. The address of the slave can be set manually via Menu P2.8. A master addresses a slave by placing the slave address in the address field of the message. When the slave sends its response, it places its own address in this address field of the response to let the master know which slave is responding. Address 0 is used for the broadcast address, which all slave devices recognize.

Function Field

The function code field of a message frame contains eight bits. Valid codes are in the range of 1 – 255 decimal. When a message is sent from a master to a slave device the function code field tells the slave what kind of action to perform. For instance, the function might be to read the data contents of a group of registers; to read the diagnostic status of the slave; to write to designated registers.

When the slave responds to the master, it uses the function code field to indicate either a normal (error-free) response or that some kind of error occurred (called an exception response). For a normal response,

the slave simply echoes the original function code. For an exception response, the slave returns a code that is equivalent to the original function code with its most-significant bit set to a logic 1.

Function Code Summary

Decimal	Hexadecimal	Description
03	0x03	Read holding registers.
04	0x04	Read input registers
06	0x06	Write/Preset
16	0x10	Actuator Control

Data Field

The data field is constructed using sets of two hexadecimal digits, in the range of 00 to FF hexadecimal. These can be made from a pair of ASCII characters, or from one RTU character, according to the network's serial transmission mode. The data field of messages sent from a master to slave devices contains additional information which the slave must use to take the action defined by the function code.

This can include items like discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the field. If no error occurs, the data field of a response from a slave to a master contains the data requested. If an error occurs, the field contains an exception code that the master application can use to determine the next action to be taken.

The data field can be nonexistent (of zero length) in certain kinds of messages. For example, in a request from a master device for a slave to respond with its communications event log (function code 0B hexadecimal), the slave does not require any additional information. The function code alone specifies the action.

Error Checking Field

Two kinds of error-checking methods are used for standard Modbus networks. When RTU mode is used for character framing, the error checking field contains a 16-bit value implemented as two 8-bit bytes. The error check value is the result of a Cyclic Redundancy Check calculation performed on the message contents.

Error Correction Mechanisms

Parity Checking: This mechanism is used for error detection on each byte. It is the simplest technique for detecting and correcting errors. The Most significant bit (MSB) of an 8-bits word is used as the parity bit and the remaining 7 bits are used as data or message bits. The parity of 8-bits transmitted word can be either even parity, odd parity or none.

Even parity means the number of 1's in the given word including the parity bit should be even while odd parity means the number of 1's in the given word including the parity bit should be odd. When the data bits are generated, an extra bit is added to satisfy the parity condition (even or odd).

CRC Checking: When RTU mode is used for character framing, the error checking field contains a 16-bit value implemented as two 8-bit bytes. The error check value is the result of a Cyclic Redundancy Check calculation performed on the message contents. The CRC field is appended to the message as the last field in the message. The low-order byte of the field is appended first, followed by the high-order byte. The CRC high-order byte is the last byte to be sent in the message.

This code detects all (single) error bursts of up to 16 bits in length. The CRC register is initialized to all '1s' and the CRC is calculated octet by octet starting with the least significant bit of 'byte 0'. The CRC calculated is bit-wise inverted and appended to the data bytes with the most significant bit transmitted earliest.

In receive side, a 16-bit CRC is generated from the data bytes of each data block as above and the bitwise inverted value is compared with the received two CRC bytes (high byte; low byte). If a miss-match is present, then an error has been detected.

Reference program in C language for CRC 16

```
unsigned int crc16 (unsigned char *buf, unsigned int length)
{
    unsigned int i;
    unsigned int j;
    unsigned int c;
    unsigned int crc = 0xFFFF;
    for (i = 0; i < length; i++)
    {
        c = *(buf + i) & 0x00FF;
        crc ^= c;
        for (j = 0; j < 8; j++)
        {
            if (crc & 0x0001)
            {
                crc >>= 1;
                crc ^= 0xA001;
            }
        }
    }
}
```

```
        else
        {
            crc >>= 1;
        }
    }
    crc = (crc >> 8) + (crc << 8);
    return (crc);
}
```

6. Modbus Communication

This sections provides a detailed view of the implementation of the Modbus RTU protocol for the Ontrac Modbus Module (OMM). The Ontrac actuator Modbus implementation is based on a simple protocol containing the slave address, a function code, the process data, and the check-sum.

Some Variable, Type, Description and Direction.

In the context of this documentation, variables refer to the content of the data field of the Modbus Telegram. The OOM Modbus RTU makes use of a number of variable types. Some variables are found in the request while some are found in the response.

Variables that are found in the request are in the IN direction while variables that are found in the response are in the OUT direction. It is important to distinguish between the IN variables and the OUT variables because their length are different. The IN variables have a length of 2 bytes while the OUT variables have a length of 7 bytes.

SetpointValue and SetpointStatus: The setpoint value and the setpoint status are variables that are used in requests, this makes them IN variables. The two variables are a WORD datatype. The setpoint value tells the actuator how far the valve should open or close. The value of the setpoint is in the range of 0% (0) to 100% (1000). The setpoint status is a BYTE datatype, it instructs the actuator to check if it is in the desired setpoint.

ReadbackValue: The datatype of the Readback Value is FLOAT. This is and OUT variable that stores the position of the current position of the actuator. The values of this variable are in the range of 0% (0) to 100% (1000).

ReadbackStatus: The Readback status specifies if the current Readback Value is valid.

Pos_D Value: This is a BYTE datatype that gives the discrete position of the actuator. The four possible values of this variable are given in the summary table below.

LocalSwitch: The LocalSwitch variable is a BYTE that stores the position of the local switch. The value of the LocalSwitch switch stores the state of the Mode Selection Switch. The possible values of this variable is given in the table below.

Variable Summary

Variable	Offset	Data Type	Telegram Type	Description
SetpointValue	0	WORD	IN	Preset valve position in from 0% (0) to 100% (1000).
SetpointVariable.	0	WORD	IN	0x8000
ReadbackValue	0	FLOAT	OUT	Current actuator position.
ReadbackStatus	5	BYTE	OUT	Status of "ReadbackValue" object.
PosDValue	6	BYTE	OUT	Discrete actuator position. 0x00 Hex: Not adjusted. 0x01 Hex: Actuator is in "CLOSED" position. 0x02 Hex: Actuator is in "OPEN" position. 0x03 Hex: Actuator is between "OPEN" and "CLOSED".
LocalSwitch	7	BYTE	OUT	Mode Selector Switch. 0x00: Mode Selector set to "O/S" 0x01: Mode Selector set to "LOCAL" 0x02: Mode Selector set to "REMOTE"

Data Structure in Internal Memory

The controller can directly read and write actuator registers using the Modbus protocol. Read-only (RO) Registers (Input registers) and Read-Write (R/W) Registers (Holding registers) are present in the actuators and they are collectively referred to as "Modbus registers." Any attempt to write a read-only (input) register or input status will generate an exception (address error).

The Modbus functions and the corresponding registers that they are used with are listed in the table below.

Address Table (Writable Register) for the Multitrac Actuator and the Function Code 10H for reading the Instruction Register (HOLDING-REGISTER).

Register Address (Hexadecimal Number)	Parameter Name	Unit	Default	Corresponding Menu Position	Meaning
110H	Torque_O			P1.1	The percentage of switch off torque for valve opening
111H	Torque_C			P1.2	The percentage of switch off torque for valve closing

112H	SPEED_O			P1.3	Percentage of valve opening speed
113H	SPEED_C			P1.4	Percentage of valve closing speed
114H	CUTOFF_O			P1.5	Switch off mode for "OPEN" direction
115H	LIMIT_O			P1.6	Switch off position for "OPEN" direction
116H	CUTOFF_C			P1.7	Switch off mode for "CLOSE" direction
117H	LIMIT_C			P1.8	Switch off position for "CLOSE" direction
120H	OPER_KEY			P2.1	Local operation button Responding mode set up
121H	Digital_INP			P2.2	Set up for remote on/off control signal response mode
122H	START_O			P2.4	Set up for increasing effective torque for "OPEN" direction
123H	START_C			P2.5	Set up for increasing effective torque for "CLOSED" direction
124H	SLAVE_ADD			P2.6	Set up Profibus address
125H	Modbus_ADD			P2.7	Set up Profibus DP with slave redundancy
150H	BE1_S			P7.1	BE1 input setting
151H	BE2_S			P7.2	BE2 input setting
152H	BE3_S			P7.3	BE3 input setting
153H	BE4_S			P7.4	BE4 input setting
154H	VALVE_1P			P7.5	Intermediate valve position indication 1
155H	VALVE_2P			P7.6	Intermediate valve position indication 2
160H	BA1_S			P8.1	BA1 setting
161H	BA2_S			P8.2	BA2 output setting
162H	BA3_S			P8.3	BA3 output setting
163H	BA4_S			P8.4	BA4 output setting
164H	BA5_S			P8.5	BA5 output setting
165H	BA6_S			P8.6	BA6 output setting
166H	BA7_S			P8.7	BA7 output setting
167H	BA8_S			P8.8	BA8output setting
280H	ANO_POL			P3.9	Analog output polarity setting
281H	ANI_SET			P3.10	Actuator's responding mode when the analog input signal is lost
282H	ANI_POS			P3.11	
283H	ANI_REF			P3.12	Analog input signal range selection
284H	ESD_POS			P3.13	ESD valve's target position setting
285H	MODE			P3.3	Control mode selection
286H	Analog Resolution			P3.8	Dead-zone adjustment
0500H	PositionSV(H)			/	Operation control setting value
0501H	PositionSV(H)			/	

Address Table (R/W) for Multitrac Actuator and the Function 03H for reading the Instruction Register (HOLDING-REGISTER)

Register Address (Hexadecimal Number)	Parameter Name	Unit	Default	Corresponding Menu Position	Meaning	R/W
110H	Torque_O			P1.1	The percentage of switch off torque for valve opening	R/W
111H	Torque_C			P1.2	The percentage of switch off torque for valve closing	R/W
112H	SPEED_O			P1.3	Percentage of valve opening speed	R/W
113H	SPEED_C			P1.4	Percentage of valve closing speed	R/W
114H	CUTOFF_O			P1.5	Switch off mode for "OPEN" direction	R/W
115H	LIMIT_O			P1.6	Switch off position for "OPEN" direction	R/W
116H	CUTOFF_C			P1.7	Switch off mode for "CLOSE" direction	R/W
117H	LIMIT_C			P1.8	Switch off position for "CLOSE" direction	R/W
120H	OPER_KEY			P2.1	Local operation button Responding mode set up	R/W
121H	Digital_INP			P2.2	Set up for remote on/off control signal response mode	R/W
122H	START_O			P2.4	Set up for increasing effective torque for "OPEN" direction	R/W
123H	START_C			P2.5	Set up for increasing effective torque for "CLOSED" direction	R/W
124H	SLAVE_ADD			P2.6	Set up Profibus address	R/W
125H	Modbus_ADD			P2.7	Set up Profibus DP with slave redundancy	R/W
150H	BE1_S			P7.1	BE1 input setting	R/W
151H	BE2_S			P7.2	BE2 input setting	R/W
152H	BE3_S			P7.3	BE3 input setting	R/W
153H	BE4_S			P7.4	BE4 input setting	R/W
154H	VALVE_1P			P7.5	Intermediate valve position indication 1	R/W
155H	VALVE_2P			P7.6	Intermediate valve position indication 2	R/W
160H	BA1_S			P8.1	BA1 setting	R/W
161H	BA2_S			P8.2	BA2 output setting	R/W
162H	BA3_S			P8.3	BA3 output setting	R/W
163H	BA4_S			P8.4	BA4 output setting	R/W
164H	BA5_S			P8.5	BA5 output setting	R/W

165H	BA6_S			P8.6	BA6 output setting	R/W
166H	BA7_S			P8.7	BA7 output setting	R/W
167H	BA8_S			P8.8	BA8 output setting	R/W
280H	ANO_POL			P3.9	Analog output polarity setting	R/W
281H	ANI_SET			P3.10	Actuator's responding mode when the analog input signal is lost	R/W
282H	ANI_POS			P3.11		R/W
283H	ANI_REF			P3.12	Analog input signal range selection	R/W
284H	ESD_POS			P3.13	ESD valve's target position setting	R/W
285H	MODE			P3.3	Control mode selection	R/W
286H	Analog Resolution			P3.8	Dead-zone adjustment	R/W
0500H	PositionSV(H)			/	Operation control setting value	R/W
0501H	PositionSV(H)			/		R/W
400H	ErrorInformation(4)			/	Equipment fault information	RO
401H	ErrorInformation(3)			/		RO
402H	ErrorInformation(2)			/		RO
403H	ErrorInformation(1)			/		RO
404H	AlarmInformation(H)			/	Equipment alarm information	RO
405H	AlarmInformation(L)			/		RO
406H	TimeOpenSet(H)			/	Equipment time for open direction	RO
407H	TimeOpenSet(L)			/		RO
408H	TimeCloseSet(H)			/	Equipment time for close direction	RO
409H	TimeCloseSet(L)			/		RO
40AH	MaxSet(H)			/	Maximum torque value	RO
40BH	MaxSet(L)			/		RO
40CH	SwRevSet(H)			/	Device software version number	RO
40DH	SwRevSet(L)			/		RO
40EH	HwRevSet(H)			/	Device hardware version number	RO
40FH	HwRevSet(L)			/		RO
410H	ActModel(H)			/	Equipment model	RO
411H	ActModel(L)			/		RO
412H	PhaseNum(H)			/	Number of equipment power supply phases	RO
413H	PhaseNum(L)			/		RO
414H	ExtModel(H)			/	Equipment expansion board model	RO
415H	ExtModel(L)			/		RO
416H	ElectrTemp(H)			/	Electronic unit temperature value	RO
417H	ElectrTemp(L)			/		RO

418H	PerNowFact(H)			/	Current valve position value 0.1%	RO
419H	PerNowFact(L)			/		RO
420H	Torque.Now			/		RO
250H	DeviceSerNumH(H)			/	Equipment serial number	RO
251H	DeviceSerNumH(L)			/		RO
252H	DeviceSerNumL(H)			/		RO
253H	DeviceSerNumL(L)			/		RO

Address Table (Read/Write) for Easytrac Actuator and the Function code 03 for reading the Instruction Register (HOLDING-REGISTER).

Parameter Address	Parameter Name	Description	Unit	Default	Description
40001	Language	Display language setting	-	0	0--English 1--Chinese 2--Russia
40002	Address	Modbus communication address, effective after restart	-	1	1-247
40003	Bps	Modbus communication baud rate, effective after restart	-	15---9600bps	12---1200bps 13---2400bps 14---4800bps 15---9600bps 16---14400bps 17---19200bps
40004	OperationKeyMode	Key operation mode	-	8--Inching mode	8--Inching mode 9--Self-holding mode
40005	DigitalInputMode	Digital input mode	-	8--Inching mode	8--Inching mode 9--Self-holding mode
40006	PhaseNumSet	Power phase number setting	-	41--Three-phase	41--Three-phase 42--Single-phase
40007	MotorSequence	Motor wiring phase sequence setting	-	43--Positive sequence	43--Positive sequence 44--Negative sequence
40008	InputResolution	Controls dead zone Settings	-	10	1-30
40009	StopSetting	Internal dead zone setting	-	20	2-50

40010	Out4mA	Output 4mA calibration	-	100	0-200
40011	Out20mA	Output 20mA calibration	-	100	0-200
40012	DI1Cfg	Configuration value of digital input port 1	-	22-Open direction indication	21-Set point enable 22-Open direction indication 23-Close direction indication 24--Stop indication 25--Alarm reset 26--ESD indication
40013	DI2Cfg	Configuration value of digital input port 2	-	23-Close direction indication	The same as above
40014	DI3Cfg	Configuration value of digital input port 3	-	24--Stop indication	The same as above
40015	DI4Cfg	Configuration value of digital input port 4	-	26--ESD indication	The same as above
40016	DI5Cfg	Configuration value of digital input port 5	-	21-Set point enable	The same as above
40017	DO1Cfg	Configuration value of digital output port 1	-	31-Full open indication	31-Full open indication 32-Full close indication 33-Remote indication 34-Fault alarm
40018	DO2Cfg	Configuration value of digital output port 2	-	32-Full close indication	The same as above
40019	DO3Cfg	Configuration value of digital output port 3	-	33-Remote indication	The same as above
40020	DO4Cfg	Configuration value of digital output port 4	-	34-Fault alarm	The same as above
40021	DO5Cfg	Configuration value of digital output port 5	-	33-Remote indication	The same as above
40022	DO6Cfg	Configuration value of digital output port 6	-	33-Remote indication	The same as above
40023	DpAddr	DP communication address	-	1	SR redundancy mode : 1-125 FR redundancy mode : 1-62
40024	ReduncyMode	DP redundancy mode	-	45--SR redundancy	45--SR redundancy 46--FR redundancy
40025	ESDMode	ESD mode	-	47-Keep current	47-Keep current 48-Target valve position
40026	EsdPosSet	ESD setting position	%	50	0-100
40027	/	/	/	/	/
40028	/	/	/	/	/
40029	/	/	/	/	/
40030	/	/	/	/	/
40031	/	/	/	/	/
40032	/	/	/	/	/

40033	/	/	/	/	/
40034	AI_Lost_Set	Analog input signal interrupt response mode	-	47-Keep current	47-Keep current 48-Target valve position
40035	AI_Target	Analog signal interrupt target valve position	%	50	0-100
40036	/	/	/	/	/
40037	/	/	/	/	/
40038	/	/	/	/	/
40039	DebugDisp	Commissioning display function switch		4--Function off	4--Function off 5--Function on
40040	MotorLineCheck	Motor line detection function switch		5--Function on	4--Function off 5--Function on
40041	/	/	/	/	/
40042	/	/	/	/	/
40043	ModBusPermilTarget	Modbus set valve position	0.1%	1000	0-1000
40044	ModBusOnOFFTarget	Modbus digital value control byte			0x03---Open 0x0C---Close 0xC0---Stop

Function code 04H and Address Table (Read Only) for reading the Easytrac Instruction Register (INPUT REGISTER).

Parameter Address	Parameter Name	Description	Unit	Default	Description
30001-30004	Equipment name			Easytrac	
30005	Valve position value (1000 when fully open, 0 when fully closed)		0.1%		Current valve position
30006	ErrorSets		-		uint16_t MotorTemp:1; // 1 Motor Temperature Fault uint16_t ElectrTemp:1; // 2 Electr Temperature Fault uint16_t EndPosition:1; // 3 End Position Fault uint16_t TorqueOpen:1; // 4 Torque Open Over Load Fault uint16_t TorqueClose:1; // 5 Torque Open Over Load Fault uint16_t BlockOpen:1; // 6 Open Block uint16_t BlockClose:1; // 7 Close Block uint16_t MotorLine:1; // 8 Motor Line Connect Error uint16_t PhaseLoss:1; // 9 Phase Loss
30007	AlarmSet		-		uint16_t Setpoint:1; // 10 Setpoint Current over 20mA
30008	Absolute valve position				0-4095

30009					
30010					
30011	Current event pointer				
30012	Event array 1				
30013	Location of event 1				
30014	Event array 2				
30015	Location of event 2				
30016	Event array 3				
30017	Location of event 3				
30018	Event array 4				
30019	Location of event 4				
30020	Event array 5				
30021	Location of event 5				
30022	Event array 6				
30023	Location of event 6				
30022	Event array 7				
30023	Location of event 7				
30022	Event array 8				
30023	Location of event 8				
30022	Event array 9				
30023	Location of event 9				
30022	Event array 10				
30023	Location of event 10				

Address Table (Writable) of Instruction Register (SINGLE COIL) and the corresponding Function code 05H. (Easytrac Actuator)

Parameter code	Parameter Name	Description	Unit	Default	Description
00000	ModbulateStart	Regulation control operation enable	-		0--Control stop 1--Control start
00001	OnOffStart	Switch control operation enable	-		0--Control stop 1--Control start

7. Actuator Control

Control of the actuator via the Modbus protocol is built on the basic idea of writing to registers and reading from register in the actuators. Any other control logic can be built on that regardless of how complex it is. Examples of Modbus request and response telegram for closing a valve connected to the actuator is provided below.

Examples of Modbus Telegrams

Objective: Master Request to Drive the valve to 100%, (This means sending a value of 1000 to the slave.)

Request: The table below breaks down the component of the request frame. We assume that the address of the actuator (slave is 01).

Operation	RTU (hex)
Slave Address	01
Function Code	10
Register Address	40 AA
We want to send a value of 1000, in HEX format this will have a value of 0x03E8 This value (0x03E8) is added to 0x8000 to give 0x83E8	
Data Value	83E8
Compute CRC using the previous algorithm	94 97
Telegram	01 10 40 AA 83 E8 94 97

Response: The table below breaks down the component of the response frame. We assume that the address of the actuator.

Operation	RTU (hex)
Slave Address	01
Function Code	10
Register Address	40 AA
Data Value	42 1B 0E E5 80 03 01

ReadbackValue: 0x421B0EE5(Float)=38.76(Dec). (The valve position is 38.76% of full stroke now) ReadbackStatus: 0x80 means valid valve position reply. PosDValue: 0x03, Actuator now is between “OPEN” and “CLOSED”. LocalSwitch: 0x01, Actuator Mode Selector set to “Local”	
Compute CRC using the previous algorithm	87 D3
Telegram	01 10 40 AA 42 1B 0E E5 80 03 01 87 D3

Reading Data from the Actuator Holding Register

Errors and alarms are also variables that can be read from the actuator. Information about the error variables are given in the table below. These variables are stored in the holding register and they can be accessed using the 03 function code.

The request PDU or telegram consists of 5 bytes while the normal response varies in length from 4 bytes up to 252 bytes depending on the number of holding registers requested.

Function Code	Variable	Data Type	Byte Number	Address
03	Actuator_Error	Unsigned Int32	4	0x0400 to 0x0403
03	Actuator_Alarm	Unsigned Int16	2	0x0404 to 0x0405

Error and Alarm Registers

The table below shows the error and alarm registers for the Multitrac Modbus Communication module. The content of the registers can be read using the 03 function. Examples of these request and functions are shown in subsequent sections.

Address	Register Content															
0x0400-0x0401	ERROR SIGNALS (0x0401)								ERRORS SIGNAL (0x0400)							
	RELAY	AUXPOWER	BLOCK_C	BLOCK_O	VOL_UNDR	VOL_OVER	IPM_FLT	TORQUE_C	TORQUE_Q	END_POS	HANDWHL	TEMP_ELC	TEMP_MOT	PHASE	BV_EXCPT	ELECTR
	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08	Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
0x0402-0x0403	ERROR SIGNALS (0x0403)								ERROR SIGNALS (0x0402)							
	---	---	---	---	---	---	---	---	---	---	---	---	ACT_INT	24_EXCPT	HALL_SEN	MOT_LINE
	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08	Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
ACT_INIT: Actuator Initialization Failure																
0x0404-0x0405	ALARM SIGNALS (0x0405)								ALARM SIGNALS (0x0402)							
	----	VOL_UNDR	VOL_OVER	AC_PHASE	ACT_TYPE	SETPOINT	LOAD_EXC	STEP/SH	CNT_TRQ	24-VOLT	TEMP_ELC	TEMP_MOT	TORQUE_C	TORQUE_O	BAT_EMPTY	BAT_CHAN
	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08	Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00

Examples of Modbus Telegram for reading Errors.

Request: The Modbus request below reads errors from the register using function code 03. We also assume that the slave address is 01.

Operation	RTU (hex)
Slave Address	01
Function Code	03
The starting address	04 00

The number of reply data	04
Compute CRC using the previous algorithm	59 86
Telegram	01 03 04 00 04 59 86

Response 1: In this reply, three error codes were returned.

Operation	RTU (hex)
Slave Address	01
Function Code	03
The starting address	04
First reply data from address 0x0400	00
Second reply data from address 0x0410	10
Third reply data from address 0x0410	80
The reply data from address 0x0403	28
Compute CRC using the previous algorithm	9A 28
Telegram	01 03 04 00 10 80 28 9A 28

Response 2: In this reply, no error codes were returned.

Operation	RTU (hex)
Slave Address	01
Function Code	03
The starting address	04
First reply data from address 0x0400	00
Second reply data from address 0x0410	00
Third reply data from address 0x0410	00
The reply data from address 0x0403	00
Compute CRC using the previous algorithm	FA 33
Telegram	01 03 04 00 10 80 28 FA 33

Response 3: Temperature limit for motor exceeded. (Error bit 3)

Operation	RTU (hex)
Slave Address	01
Function Code	03
The starting address	04
First reply data from address 0x0400	00
Second reply data from address 0x0410	00
Third reply data from address 0x0410	00
The reply data from address 0x0403	08
Compute CRC using the previous algorithm	FB F5
Telegram	01 03 04 00 00 00 08 FB F5

Error Response 4: Blocking in open direction. (Error bit 12)

Operation	RTU (hex)
Slave Address	01
Function Code	03
The starting address	04
First reply data from address 0x0400	00
Second reply data from address 0x0410	00
Third reply data from address 0x0410	10
The reply data from address 0x0403	00
Compute CRC using the previous algorithm	F7 F3
Telegram	01 03 04 00 00 10 00 F7 F3

Error Response 5: Blocking in close direction. (Error bit 12)

Operation	RTU (hex)
-----------	-----------

Slave Address	01
Function Code	03
The starting address	04
First reply data from address 0x0400	00
Second reply data from address 0x0410	00
Third reply data from address 0x0410	20
The reply data from address 0x0403	00
Compute CRC using the previous algorithm	E3 F3
Telegram	01 03 04 00 00 20 00 E3 F3

Examples of Modbus Telegram for reading Alarms

Request: The Modbus request below reads errors from the register using function code 03. We also assume that the slave address is 01.

Operation	RTU (hex)
Slave Address	01
Function Code	03
The starting address	04 04
The number of reply data	02
Compute CRC using the previous algorithm	DB 44
Telegram	01 03 04 04 02 DB 44

Alarm Response 1: Ac Phase Setting Error

Operation	RTU (hex)
Slave Address	01
Function Code	03
The number of reply data	02
0001000 from address 0x0405	10

0000000 from address 0x0404	00
Compute CRC using the previous algorithm	B5 84
Telegram	01 03 02 10 00 B5 84

Alarm Response 2: Battery Empty.

Operation	RTU (hex)
Slave Address	01
Function Code	03
The number of reply data	02
0001000 from address 0x0405	00
0000000 from address 0x0404	02
Compute CRC using the previous algorithm	39 85
Telegram	01 03 02 00 02 39 85

Alarm Response 2: No alarm

Operation	RTU (hex)
Slave Address	01
Function Code	03
The number of reply data	02
0001000 from address 0x0405	00
0000000 from address 0x0404	00
Compute CRC using the previous algorithm	B5 84
Telegram	01 03 02 00 00 B8 44

8. Device Configuration

Actuator and Modbus Configuration

Smartrac is a PC based system for digitally communicating directly to actuators. It will provide users with an Engineer level access to all of the settings and data that the actuator contains. Smartrac requires a standard PC (Windows 10 or above) and a dedicated RS-485 standard cable.

The necessary parameters that are required for the communication can be configured on the through the Smartrac. A detailed information on the use of the smartrac software can be found in the smartrac manual.



Figure 18 Smartrac Software Interface

9. Troubleshooting

Generally, most of the problems associated with Modbus enabled actuators are caused by wiring errors and are simple to test and fix. One of the things to check for is the wiring. The technician needs to confirm if the network cable is connected to the correct actuator terminal.

The correct wiring diagram can be obtained from the Cloudtrac. Every Ontrac actuator comes with a permanent QR code attached. Simply scanning the QR (Quick Response) code on your actuator connects you to Cloudtrac. The cloud provides access to the specific information for your actuator.

Broken cables are also sources of errors, this will prevent signals from reaching the device and it will also cause undesired effect such as noise and reflection. Any form of debugging that involves swapping cables or changing any electrical connection should be carried out only when the actuator is powered down .

If communication issues persist after the electrical connections have been verified, please confirm that the correct address, baud rate and parity have been set in the actuator and that they correspond with the settings in the known working actuators.

If the communication issues remain, please contact Ontrac Technical Support team through the support page on the website (www.ontracactuators.com). The support agent will request for information about the actuator and provide the necessary assistance.