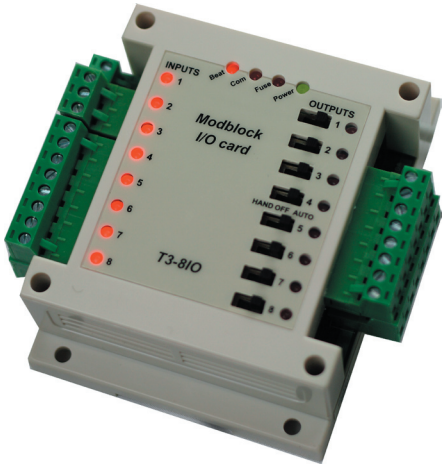
T3 Modules

User's Manual



Contents

1 Introduction

1.1 About this manual

The purpose of this manual is to provide the instructions to simply and quickly install and operate the T3 Module equipment. The manual begins with a general description of the product followed by the instructions for a correct hard-ware installation. Its configuration and operation of the device are later described in detail.

1.2 General Description of the Product

The T3 Series are general purpose input / ouput modules for building integrators. Available in several input/output configurations, the T3 Series modules provide convenient termination for field devices and interfacing to your HVAC, lighting, temperature sensors, and other typical building automation applications. Each of the analog inputs can be jumper configured for signals of either 0-5V, 0-20mA, or dry contact. The outputs are available in dry contacts 1amp/output, 0-10V analog, and PNP sinking. The modules are slave devices that can be easily controlled via the RS485 serial interface using the industry standard Modbus Protocol.

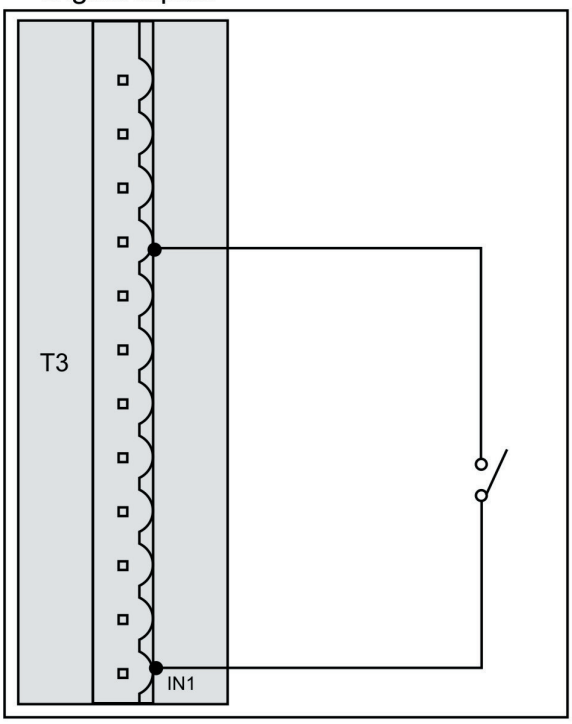
Highlights: -Surge-protected analog inputs with 10-bit resolution.
-Outputs can individually be switched to ON, OFF, AUTO.
-High impact plastic enclosure provides durability in com-mercial environments.
-Standard modbus protocol allows for up to 254 unique de-vices on one RS485 network.

1.3 Technical Data

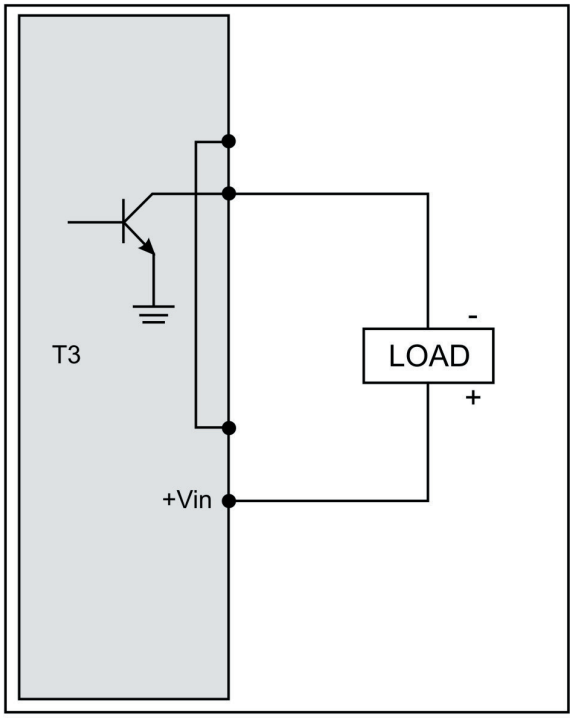
T3-8IO-A ..... 8 analog outputs @0-10VDC 200mA total, 8 analog inputs @0-5V, 0-20mA, dry
T3-32I ..... 32 analog inputs @0-5V, 0-10V, 0-20mA, dry
T3-8I16O ..... 16 PNP Sinking outputs @12V 200mA total, 8 analog inputs @0-5V, 0-20mA, dry
T3-8I13O ..... 13dry-contact relay outputs x1amps @120V, 8 analog inputs @0-5V, 0-20mA, dry
Temperature range ……..…..... -40~100°C (-40~212°F)
Supply voltage ……..…..... 24VAC ±20%, 50-60Hz
Power consumption ...…..…….. 100mA at 12VDC
Relay contacts rating ..…….....…... max 1A
Ambient temperature: Operation……. 0~70°C (32~158°F)
Storage……. 2-50°C (35~120°F)
Ambient humidity…….....…..... 10-90 %Rh
Material, enclosure…….....…..... Flame proof plastic
Enclosure rating……..... IP31
Temperature sensor…….....…..... 10K thermistor ±0.5°C
Colour……...…..... White/Off-white

1.4 Wiring Diagram

Digital Inputs

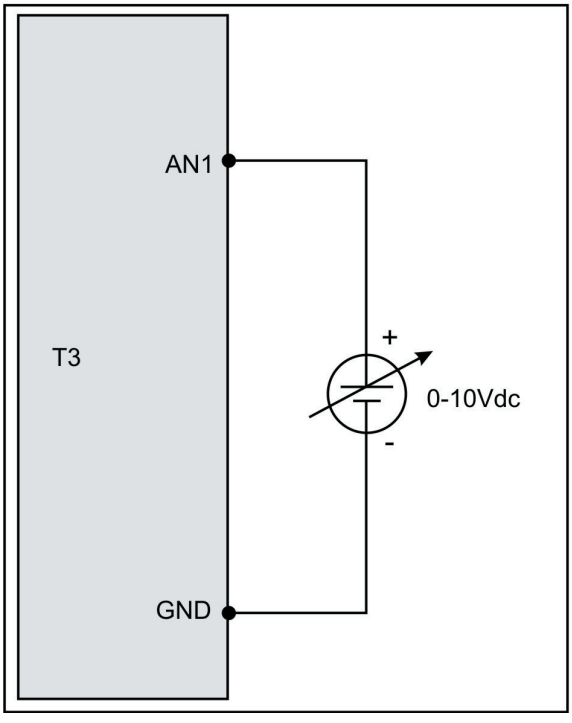


Digital Outputs

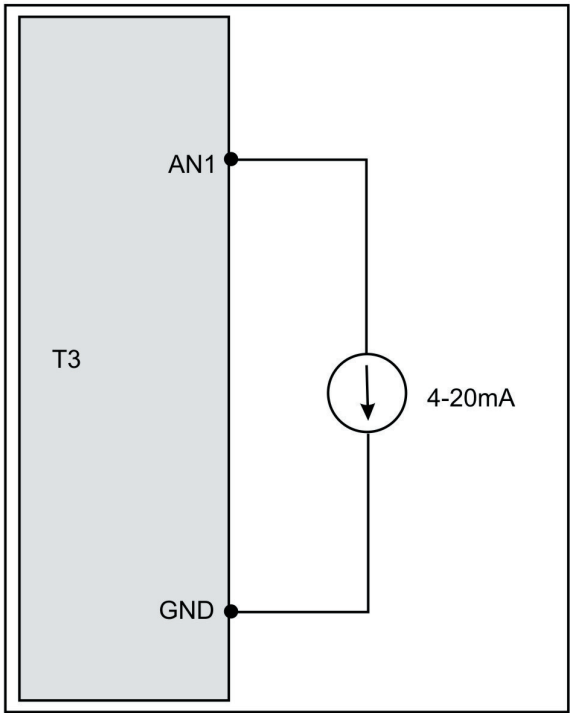


Analog Inputs

Voltage 0-10V



Current 4-20mA



1.5 Standard Operation

1.5.1 Inputs

Each input of a T3 module can be jumper-configured in 1 of 3 ways: •0-5V signal
•0-20mA signal
•Dry contact, thermistor

The value of each input is stored as a 10-bit number in the respective modbus register.
The registers addresses are as follows:

T3 Model	Number of Inputs	Register Addresses
T3-8IO-A	8	108-115
T3-8IO-D	8	118-133
T3-32I	32	100-131
T3-8I16O	8	108-115
T3-8I13O-D	8	118-133

Table1: Input Register Addresses

A 5V, or 20mA, would give a reading of 1024. Each input has a corresponding LED which will light up if the value of the input is greater than 512. For more info on reading the input registers, see the section on Serial Communications.

1.5.2 Outputs

The state of each output is determined by its corresponding switch position. The switches have 3 states – 'hand', 'off', and 'auto'. When switched to 'hand', the corresponding output will be switched on - 10V for analog, contacts closed for relay, or 0V for sinking outputs. When switched to 'off', the output will be set to 0V for analog, open contact for relay, or open circuit for sinking outputs. When switched to 'auto', Analog outputs will be set to the level stored in the corresponding MODBUS output registers. For Digital or Sinking outputs, a register value 0 is de-activate and register value 1000 is activated.

The output registers are as follows:

T3 Model	Number of Outputs	Register Addresses
T3-8IO-A	8	100-107
T3-8IO-D	8	100-107
T3-32I	0	--
T3-8I16O	16	100-107 & 116-123
T3-8I13O-D	13	100-112

Table 2: Output Register Addresses

These registers can be changed using the RS485 serial interface. For analog outputs, a 0 corresponds to 0V. Likewise, a 1024 corresponds to 10V. For relay or sinking outputs, the output will be activated by any number greater than 512. The output registers are stored in RAM, thus the contents of each register will be lost upon power-off. Each output has a corresponding LED which will light up if the value of the output is greater than 512 (5V). For more info on writing the output registers, see the section on Serial Communications.

1.5.3 Analog Output Calibration

The T3-8IO-A has an output calibration feature that allows for an adjustment of +/- 1.28V. Calibration is controlled via the calibration register located at register address 13. By default, this is 128, which corresponds to 0V calibration. A value of 0 would give a –1.28V offset. A value of 255 would give a +1.28V offset. It is recommended that the calibration be determined while the output is set to 5V. The calibration value is located in flash memory and will be restored upon power-up.

1.5.4 Bandrate

All T3-modules have adjustable Baudrates set by MODBUS register 15. By default baud is set to 19.2kbps.
Value 1 will set the baud to 19200 bps.
Value 0 will set the baud to 9600 bps.

1.6 Master Timer Clock Function for Tstat in a Network

The T3 series can act as a master timeclock for the tstat network to set a series of tstats to occupied and unoccupied mode. The system works by connecting an ordinary mechanical timeclock or a separate controller to input#1 of the network controller. Whenever the timeclock contact opens or closes, a message is sent from the network controller out to the tstats to go into occupied and unoccupied mode.

- Opening the contact connected to input #1 of the T3 signals an occupied event, the network controller will send an occupied command to each tstat in the network. This command is sent only once to each tstat so that the user in the room can change the fan speed manually.

- Closing the contact on input #1 will signal an unoccupied event, all thermostats in the network are set to unoccupied mode. This command is sent to each tstat only once so that the local user has manual override control.

The T3 network controller will maintain a list of all tstats that are successfully commanded for each timeclock event so that each timeclock event is transmitted to each tstat one time. In this way, the users in the rooms will have local control between timeclock events. If a tstat happens to be offline, the T3 will repeat the event command until the tstat comes back online and a response is received. The T3 polls each tstat and waits approximately 1 second for a response, starting from address #1 and on up to #254. Below is a typical wiring diagram for a Master timeclock and several tstats connected on the RS485 network.

Take T3-8I8O as an example:

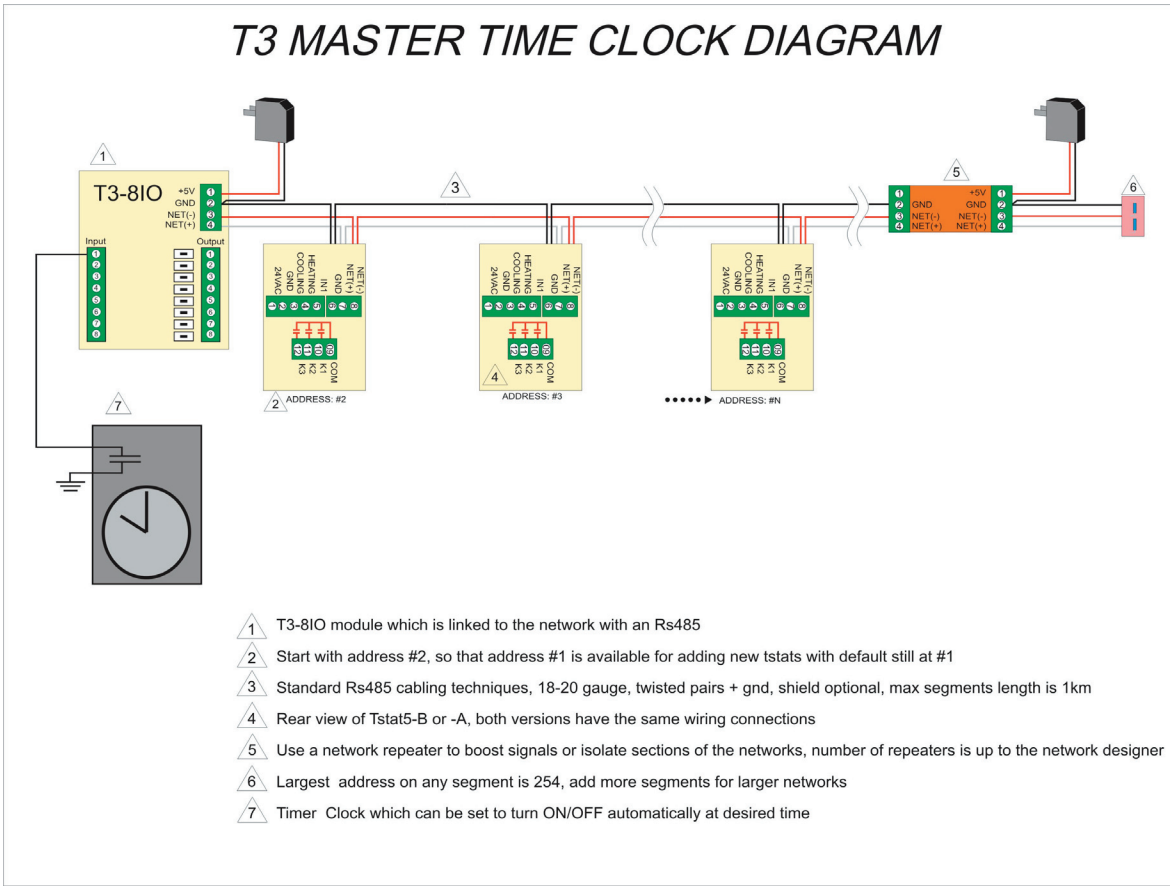
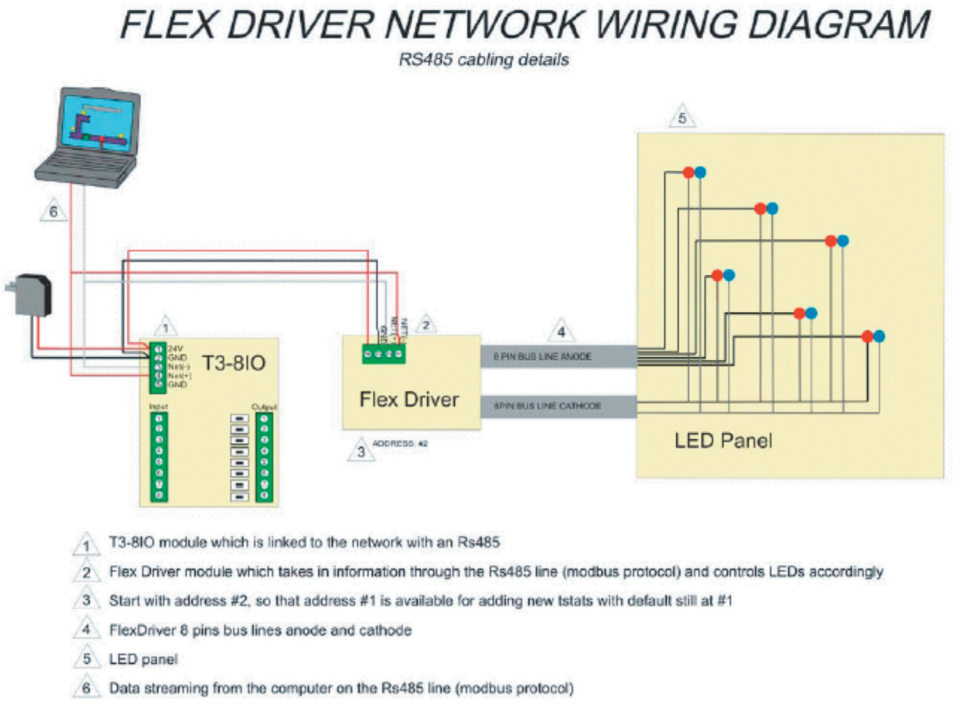


Fig. 2: Flex Driver Network Wiring Diagram

The T3 series can act as a master for the FlexDriver as well. Given the FlexDriver is only a Slave, the multi-purpose T3 is used at a medium to talk to the FlexDriver device. It can receive data from other modules and translate the information into stream of data which the FlexDriver can understand.

Take T3-8I8O as an example:



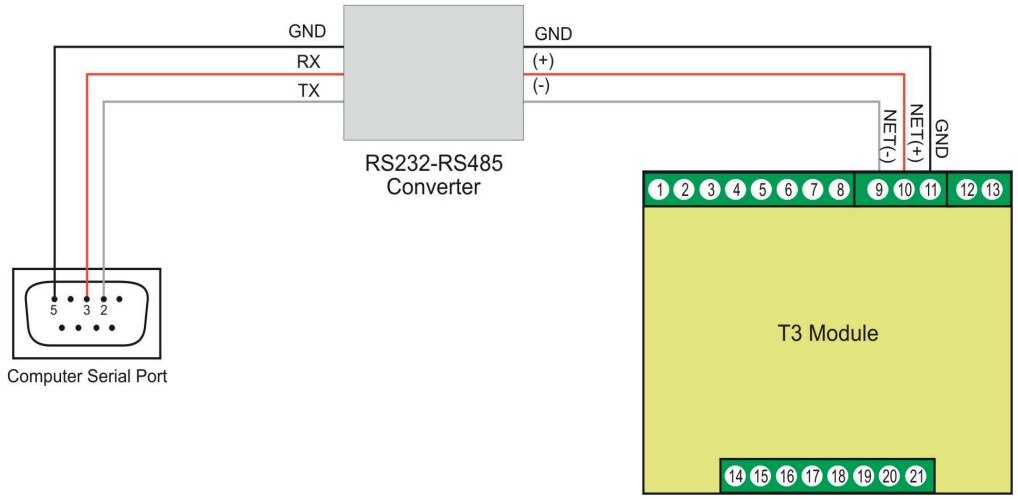
1.7 Accessing T3 Series Registers Via Serial Communications

The T3 modules have a built-in serial interface for communication over an RS485 network. Communication is currently implemented using Modbus Protocol. However, future versions of the T3 modules will work with both BACnet and TCP/IP Protocols. For detailed information on Modbus Protocol, see the chapter entitled Modbus Serial Communications.

1.7.1 Connecting the T3 module to a computer

The T3 modules connect to a computer serially via the RS485 interface. An RS232 to RS485 converter is required in order to communicate with a standard PC. Figure 14 shows how the T3 module should be connected to the serial port of a PC.

T3- COMPUTER CONNECTION



1.7.2 List of registers in the T3-8IO-A

Note: When using the Modbus Poll software, addressing should be set to “Protocol Addresses (Base 0)” under the “Display” menu.

Address	Bytes	Register and Description
0 to 3	4	Serial Number, 4 byte value
4	1	EEPROM hardware Version Number
5	1	Firmware Version Number
6	1	ADDRESS. Modbus device address
7	1	Product Model
8	1	Hardware Revision
9	1	PIC Version Number
13	1	Calibration register – used to calibrate the outputs.
15	1	Baudrate setting: 0 will set 9600bps, 1 will set 19200bps
100	2	Output 1 Register
101	2	Output 2 Register
102	2	Output 3 Register
103	2	Output 4 Register
104	2	Output 5 Register
105	2	Output 6 Register
106	2	Output 7 Register
107	2	Output 8 Register
108	2	Input 1 Register
109	2	Input 2 Register
110	2	Input 3 Register
111	2	Input 4 Register
112	2	Input 5 Register
113	2	Input 6 Register
114	2	Input 7 Register
115	2	Input 8 Register
116	2	Switch Bank 1 Register
117	2	Switch Bank 2 Register
118 - 125	1	Range for each input, 118 correspond to ch1.0 = raw data,1 = 10K Celsius, 2 = 10K Fahrenheit ,3 = 0 - 100%,4 = ON/OFF, 5 = OFF/ON

Example: register 118 reads 5 (hex 05)
register 119 reads 138 (hex 8A)
The Pulse Count for Channel1 is then 1418 pulse (hex 058A)

Writing to register 134 will clear registers 118 and 119. Subsequent registers 135 to 138 are optional memory to store date and time at which Pulse Counts have been cleared.

1.7.3 Lits of Registers in the T3-32I

Note: When using the Modbus Poll software, addressing should be set to “Protocol Addresses (Base 0)” under the “Display” menu.

Address	Bytes	Register and Description
0 to 3	4	Serial Number, 4 byte value
4	1	EEPROM hardware Version Number
5	1	Firmware Version Number
6	1	ADDRESS. Modbus device address
7	1	Product Model
8	1	Hardware Revision
9	1	PIC Version NUmber
13	1	Calibration register – used to calibrate the outputs.
15	1	Baudrate setting: 0 will set 9600bps, 1 will set 19200bps
100	2	Input 1 Register
101	2	Input 2 Register
102	2	Input 3 Register
103	2	Input 4 Register
104	2	Input 5 Register
105	2	Input 6 Register
106	2	Input 7 Register
107	2	Input 8 Register
108	2	Input 9 Register
109	2	Input 10 Register
110	2	Input 11 Register
111	2	Input 12 Register
112	2	Input 13 Register
113	2	Input 14 Register
114	2	Input 15 Register
115	2	Input 16 Register
116	2	Input 17 Register
117	2	Input 18 Register
118	2	Input 19 Register
119	2	Input 20 Register
120	2	Input 21 Register
121	2	Input 22 Register
122	2	Input 23 Register
123	2	Input 24 Register
124	2	Input 25 Register
125	2	Input 26 Register
126	2	Input 27 Register
127	2	Input 28 Register
128	2	Input 29 Register
129	2	Input 30 Register
130	2	Input 31 Register
131	2	Input 32 Register
228-259	1	Range for each input, 228 correspond to ch1.0 = raw data, 1 = 10K Celsius, 2 = 10K Fahrenheit ,3 = 0 - 100%,4 = ON/OFF, 5 = OFF/ON

1.7.4 List of Registers in the T3-8-16

Note: When using the Modbus Poll software, addressing should be set to “Protocol Addresses (Base 0)” under the “Display” menu.

Address	Bytes	Register and Description
0 to 3	4	Serial Number, 4 byte value
4	1	EEPROM hardware Version Number
5	1	Firmware Version Number
6	1	ADDRESS. Modbus device address
7	1	Product Model
8	1	Hardware Revision
9	1	PIC Version Number
13	1	Calibration register – used to calibrate the outputs.
15	1	Baudrate setting: 0 will set 9600bps, 1 will set 19200bps
100	2	Output 1 Register
101	2	Output 2 Register
102	2	Output 3 Register
103	2	Output 4 Register
104	2	Output 5 Register
105	2	Output 6 Register
106	2	Output 7 Register
107	2	Output 8 Register
108	2	Output 1 Register
109	2	Output 2 Register
110	2	Output 3 Register
111	2	Output 4 Register
112	2	Output 5 Register
113	2	Output 6 Register
114	2	Output 7 Register
115	2	Output 8 Register
117	2	Output 9 Register
118	2	Output 10 Register
119	2	Output 11 Register
120	2	Output 12 Register
121	2	Output 13 Register
122	2	Output 14 Register
123	2	Output 15 Register
124	2	Output 16 Register
125	2	Switch Bank 1 Register
126	2	Switch Bank 2 Register
128-135	1	Range for each input, 128 correspond to ch1.0 = raw data, 1 = 10K Celsius, 2 = 10K Fahrenheit ,3 = 0 - 100%,4 = ON/OFF, 5 = OFF/ON

1.7.5 List of Registers in the T3-8I13O

Address	Bytes	Register and Description
0 to 3	4	Serial Number, 4 byte value
4	1	EEPROM hardware Version Number
5	1	Firmware Version Number
6	1	ADDRESS. Modbus device address
7	1	Product Model
8	1	Hardware Revision
9	1	PIC Version Number
13	1	Calibration register – used to calibrate the outputs.
15	1	Baudrate setting: 0 will set 9600bps, 1 will set 19200bps
16 - 99	1	Reserved
100	2	Output 1 Register
101	2	Output 2 Register
102	2	Output 3 Register
103	2	Output 4 Register
104	2	Output 5 Register
105	2	Output 6 Register
106	2	Output 7 Register
107	2	Output 8 Register
108	2	Output 9 Register
109	2	Output 10 Register
110	2	Output 11 Register
111	2	Output 12 Register
112	2	Output 13 Register
113 - 115	1	Reserved
116	2	Switch Bank 1 Register
117	2	Switch Bank 2 Register
118	2	IN1 high word
119	2	IN1 low word
120	2	IN2 high word
121	2	IN2 low word
122	2	N3 high word
123	2	IN3 low word
124	2	IN4 high word
125	2	IN4 low word
126	2	IN5 high word
127	2	IN5 low word
128	2	IN6 high word
129	2	IN6 low word
130	2	IN7 high word
131	2	N7 low word
132	2	IN8 high word
133	2	IN8 low word
		Clearing Pulse Number Registers: Writing to their respective Year registers (134 for ch1, 139 for ch2, 144 for ch3...) will clear the above pulse numbers

1.7.5 List of Registers in the T3-8I13O (continued)

Address	Bytes	Register and Description
134-138	5	Date stamp of Channel 1: Year, Month, Day, Hour, Minute respectively.
139-143	5	Date stamp of Channel 2: Year, Month, Day, Hour, Minute respectively
144-148	5	Date stamp of Channel 3: Year, Month, Day, Hour, Minute respectively.
149-153	5	Date stamp of Channel 4: Year, Month, Day, Hour, Minute respectively.
154-158	5	Date stamp of Channel 5: Year, Month, Day, Hour, Minute respectively.
159-163	5	Date stamp of Channel 6: Year, Month, Day, Hour, Minute respectively.
164-168	5	Date stamp of Channel 7: Year, Month, Day, Hour, Minute respectively.
169-173	5	Date stamp of Channel 8: Year, Month, Day, Hour, Minute respectively.
174	1	Assign each channel sampling type. 0 = analog, 1 = pulse. channel 1 correspond to bit0 and ch2 correspond to bit1 and so on.
175-182	2	Analog reading from each channel, whatever the channel be set as analog or pulse mode. 175 correspond to ch1
183-190	1	Range for each input, 183 correspond to ch1.0 = raw data, 1 = 10K Celsius, 2 = 10K Fahrenheit ,3 = 0 - 100%,4 = ON/OFF, 5 = OFF/ON
191	1	Filter coefficient for input 1,0 through 100,default is 20.
192	1	Filter coefficient for input 2,0 through 100,default is 20.
193	1	Filter coefficient for input 3,0 through 100,default is 20.
194	1	Filter coefficient for input 4,0 through 100,default is 20.
195	1	Filter coefficient for input 5,0 through 100,default is 20.
196	1	Filter coefficient for input 6,0 through 100,default is 20.
197	1	Filter coefficient for input 7,0 through 100,default is 20.
198	1	Filter coefficient for input 8,0 through 100,default is 20.
199	1	Timer for input 1,how long time the lightingcontrol take over the outputs
200	1	Timer for input 2,how long time the lightingcontrol take over the outputs
201	1	Timer for input 3,how long time the lightingcontrol take over the outputs
202	1	Timer for input 4,how long time the lightingcontrol take over the outputs
203	1	Timer for input 5,how long time the lightingcontrol take over the outputs
204	1	Timer for input 6,how long time the lightingcontrol take over the outputs
205	1	Timer for input 7,how long time the lightingcontrol take over the outputs
206	1	Timer for input 8,how long time the lightingcontrol take over the outputs
207	1	Input1 timer Left,how much time left for the lighting control
208	1	Input 2 timer Left,how much time left for the lighting control
209	1	Input 3 timer Left,how much time left for the lighting control
210	1	Input 4 timer Left,how much time left for the lighting control
211	1	Input 5 timer Left,how much time left for the lighting control
212	1	Input 6 timer Left,how much time left for the lighting control
213	1	Input 7 timer Left,how much time left for the lighting control
214	1	Input 8 timer Left,how much time left for the lighting control
215	1	light control disable/enable ,each bit correspond to one output,output1 correspond to least significant bit, 0 = disable,1 = enable
216	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
217	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
218	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
219	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
220	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
221	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
222	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
223	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2

1.7.5 List of Registers in the T3-8I13O (continued)

Address	Bytes	Register and Description
224	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
225	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
226	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
227	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2
228	1	Select which input as lighting control trigger,0 = disable lighting control,1= input1,2=input2

For example, if we would like to read the input 2 register at module node address #1,

Slave Address	Function	Starting Address Hi	Starting Address Lo	No. of Points Hi	No. of Points Lo	CRC Hi Byte	CRC Hi Byte
1	3	0	109	0	1	xx	xx

Or we read 8 values after input 2 in module 1,

Slave Address	Function	Starting Address Hi	Starting Address Lo	No. of Points Hi	No. of Points Lo	CRC Hi Byte	CRC Hi Byte
1	3	0	109	0	8	xx	xx

Or we write 600 to output 4 in module 1,

Slave Address	Function	Starting Address Hi	Starting Address Lo	No. of Points Hi	No. of Points Lo	CRC Hi Byte	CRC Hi Byte
1	6	0	103	0	600	xx	xx

More details can be found in Modbus Secial Communication Section below.

1.7.6 Note about registers when updating the firmware

There are two registers that will tell the CPU information about the model and hardware of the T3 module. NOTE: after updating the firmware you MUST setup these registers first or the module may not function properly.

Product Model is register address 7. The corresponding values are as follows:
T3-8IO-A = 20
T3-8IO-D = 21
T3-32I = 22
T3-8-16 = 23

Hardware revision is register address 8. The hardware revision can be found by removing the front cover of the module. It is written in white silkscreen on the edge of the board.

1.8 Lighting Control with the T3-8I-13O module

It is possible to use the T3 module as a lighting control module, the logic is embedded directly in the T3 module to make light switching response faster and reduce the polling required for lighting applications. The general idea is that each input can be configured as a lighting switch input, any or all of the inputs can be configured this way. Each input corresponds to one lighting zone. Next any or all of the outputs can be assigned to any of the zones. And finally, each zone has a timer which sets how long the zone will go to occupied mode. An additional set of timers show how much time is left for a particular override event, these are read/write registers so that the master can also initiate events and override events under way. Finally, there is an auto/manual bit for each output to override the local T3 logic such as during commissioning or special events.

1. The “range” for each input is configured in registers 183 thru 190,
   set any input to act as a lighting switch by setting the range to 7.
   Each input corresponds to one zone,
   Short circuit the input to GND and this will trigger an event
   If the zone was previously on, it will flip to off and vice versa.

2. Each Zone has an ‘override time’ setting which sets how long the lights will be triggered on for a particular hit of the switch.
   The values are in minutes, they are read/write and are stored in registers 199 thru 206 ,
   199 is the time for zone 1 which is controlled by input1
   200 for zone2 and so on..

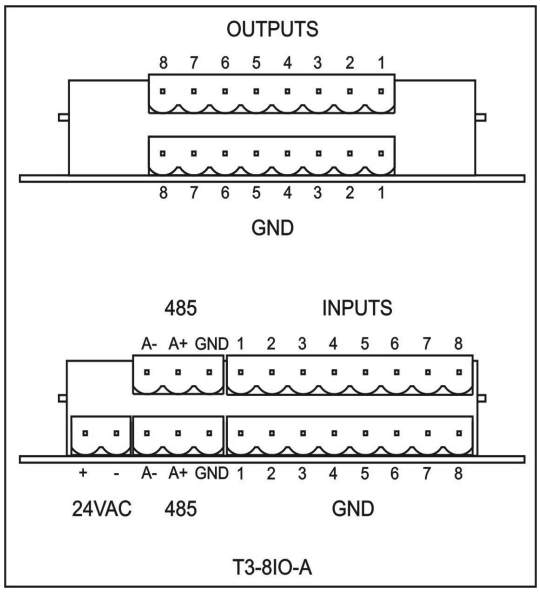
3 Each zone has a ‘time left’ register which shows the remaining time left after a particular hit on the switch.
   These are read/write values in ‘minutes’ and are stored in registers 207 thru 214
   Each time there is a hit on a particular hit on a light switch, the ‘time left’ register will be filled in with this ‘over ride time’ setting.
   For example, a hit on switch 1 will trigger a copy of register 199 to register 207.
   Then register 207 will start counting down.

4 Each output has an auto/manual bit so that the lighting control logic (and any other future logic embedded in the module) can be disabled.
   Register 215 , auto/manual register, 2 byte length.
   0 = manual (lighting control disabled),
   1 = auto (lighting control enabled).
   Each bit corresponds to one output with output 1 starting at the least significant bit
   Output 13 corresponds to the 13th bit.

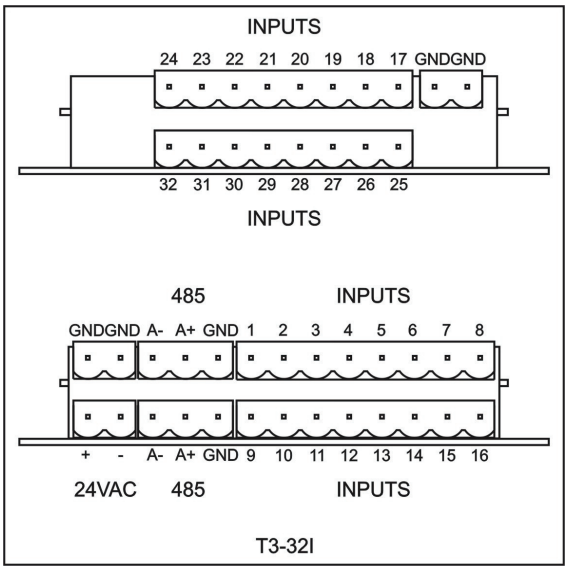
5 Assign Outputs to Zones in registers 216 thru 228
   13 outputs, one register for each output which assigns that particular output to a particular zone.
   Since there are 8 zones, the these registers will accept a value from 1 to 8,
   0 = n/a,
   1 = means this output will be linked to zone1 (and controlled by input1).
   2 = means this output will be linked to zone2 (and controlled by input2).
   And so on.

1.9 Installation

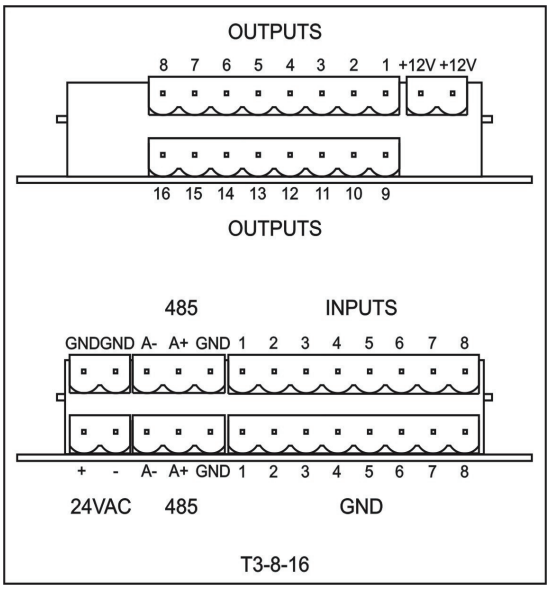
1.9.1 Terminal Block Connections (T3-8IO-A)



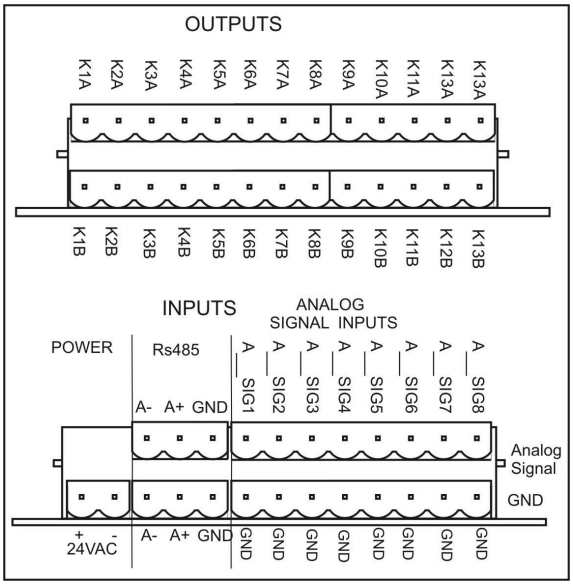
1.9.2 Terminal Block Connections (T3-32I)



1.9.3 Terminal Block Connections (T3-8-16)



1.9.4 Terminal Block Connections (T3-8-13)

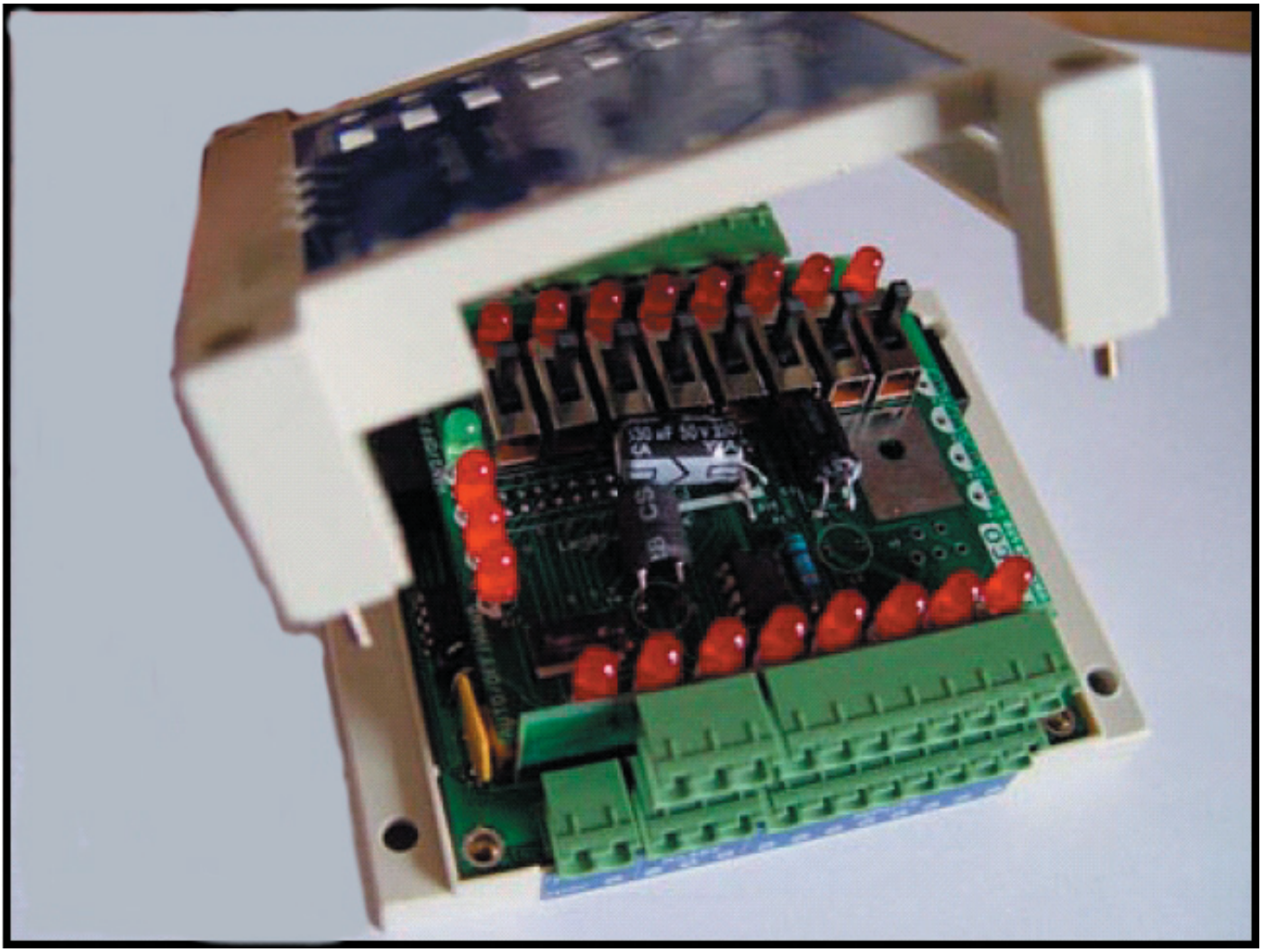


1.9.5 Mounting

External wiring is connected to a terminal block on the cir-cuit board.

The enclosure comprises a base section and a cover. The base section can be mounted directly on a wall or on a wall box.

Length of cables : Max 200m area 0.5mm2



2 Modbus Serial Communications

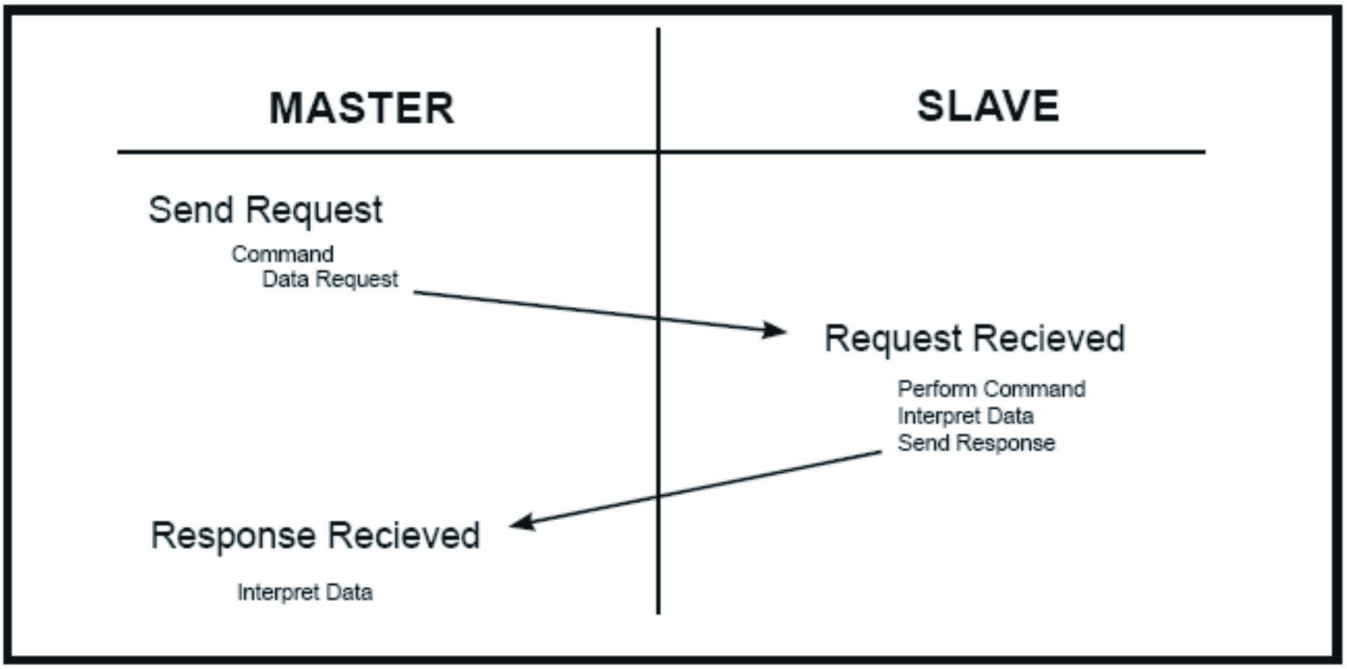
2.1 Overview

Modbus protocol is a widely used and well-documented communications method. It provides a simple and effective means of programming our various products.

A typical Modbus packet looks like this:

Byte1 Device ID, the destination address for a particular message
Byte2 Function
Byte3 Starting address of the particular storage registers) to be read or written, hi byte,
Byte4 Starting address low byte
Byte5 No. of registers to read/write (hi byte)
Byte6 No. of registers to read/write (low byte)
Byte7 CRC hi byte
Byte8 CRC low byte

During normal operation, the slave will immediately send a response to the master request.



[Notice]: Most errors during message transfer are timeout errors. This is because bytes being distorted or missing will not trigger a response resulting in a timeout error.

Software tools can be found at: http://www.modbustools.com/modbus_poll.asp If your application can read & write bytes to a separate PC running the 'Modbus Slave' application, you will be able to read & write bytes to theTstat5. Note: When using the Modbus Poll software, addressing should be set to "Protocol Addresses (Base 0)" under the "Display" menu.

2.2 Modbus Examples

2.2.1 READ Command (0x03)

This function is used to read the contents of multiple memory registers. The master to the Modbus must specify, the device ID, it's starting register and quantity of register desired. By convention if a data were to contain 2 byte, we would first send the Hi byte and then the Lo byte.

The master to the Modbus network will issue a read command:

- Device ID=11
- Read 6 bytes of data
- Starting at register number 107 (6Bh)

Byte #	Field Name (Hex)	Data	Description
Byte1	Slave Address	11	Tstat with ID11 will be read
Byte2	Function	03	Read operation
Byte3	Starting Address Hi	00	
Byte4	Starting Address Lo	6B	Reading starting from register #6B
Byte5	No. of Register to read Hi	00	
Byte6	No. of Register to read Lo	03	Read a total of 3 registers
Byte7	Error Check (CRC) HI byte	XX	The CRC is calculated using the CRC
Byte8	Error Check (CRC) LO byte	XX	routine described below

The slave device with ID=11 will answer the master within a few milliseconds with the following response.

Byte #	Field Name (Hex)	Data	Description
Byte1	Slave Address	11	Slave with ID11 is responding
Byte2	Function	03	we're responding to a read command
Byte3	Byte Count	06	6 bytes are coming
Byte4	Data1 Hi	02	byte1 of the data
Byte5	Data1 Lo	2B	byte2 of the data
Byte6	Data2 Hi	00	byte3 of the data
Byte7	Data2 Lo	00	byte4 of the data
Byte8	Data3 Hi	00	byte5 of the data
Byte9	Data3 Lo	64	byte6 of the data
Byte10	Error Check (CRC) HI byte	XX	The CRC is calculated using the CRC
Byte11	Error Check (CRC) LO byte	XX	routine described below

Example of the Read Command:
The Master sends the Read querie:

Slave Address	Function	Starting Address Hi	Starting Address Lo	No. of Points Hi	No. of Points Lo	CRC Hi Byte	CRC Lo Byte
11	3	0	(6Bh) 107	0	3	xx	xx

The device node sends back the following response:

Slave Address	Function	Byte Count	Data1 Hi	Data1 Lo	Data2 Hi	Data2 Lo
11	3	6	(02h) 2	(2Bh) 43	(00h) 0	(00h) 0

Data3 Hi	Data3 Lo	CRC Hi Byte	CRC Lo Byte			
(00h) 0	(64h) 100	xx	xx			

2.2.2 WRITE command (0x06)

This function is used to write to a single memory register. The master of the Modbus must specify the device ID, its register address to be written and the data desired.

The master to the Modbus network will issue a write command:

- Device ID=11
- Write to address 11
- Enter data 3 (03h)

Byte #	Field Name (Hex)	Data	Description
Byte1	Slave Address	11	destination address
Byte2	Function	06	this is a write command
Byte3	Register Address Hi	00	address which will be written to, hi byte
Byte4	Register Address Lo	01	address which will be written to, low byte
Byte5	Data Hi	00	data that we are writing, hi byte
Byte6	Data Lo	03	data we are writing, low byte
Byte7	Error Check (CRC) HI byte	XX	The CRC is calculated using the CRC
Byte8	Error Check (CRC) LO byte	XX	routine described below

The slave device with ID=11 will answer the master within a few milliseconds with the following response.

Byte #	Field Name (Hex)	Data	Description
Byte1	Slave Address	11	destination address
Byte2	Function	06	this is a write command
Byte3	Register Address Hi	00	address which will be written to, hi byte
Byte4	Register Address Lo	01	address which will be written to, low byte
Byte5	Data Hi	00	data that we are writing, hi byte
Byte6	Data Lo	03	data we are writing, low byte
Byte7	Error Check (CRC) HI byte	XX	The CRC is calculated using the CRC
Byte8	Error Check (CRC) LO byte	XX	routine described below

[Notice]: In this case the Slave device just sends back the message to let the Master know the query has been properly received.

Example of the Write Command

The Master sends the Write querie:

Slave Address	Function	Starting Address Hi	Starting Address Lo	Data Hi	Data Lo	CRC Hi Byte	CRC Lo Byte
11	6	0	(01h) 1	0	3	xx	xx

The device node sends back the following response:

Slave Address	Function	Starting Address Hi	Starting Address Lo	Data Hi	Data Lo	CRC Hi Byte	CRC Lo Byte
11	6	0	(01h) 1	0	3	xx	xx

2.2.3 MULTIPLE-WRITE Command (0x10)

This function is used to write to multiple memory registers. The master of the Modbus must specify the device ID, its starting address register, the amount of register desired and the data. NOTE: This is used for firmware update only. It is not used to write device registers.

The master to the Modbus network will issue a multiple-write command:

- Device ID=11
- Write to address 291 (123h)
- Number of Registers 3
- Data 1 = 10 (000Ah)
- Data 2 = 11(000Bh)
- Data 3 = 12 (000Ch)

Byte #	Field Name (Hex)	Data	Description
Byte1	Slave Address	11	destination address ID 11
Byte2	Function	10	this is a multiple write command
Byte3	Register Start Address Hi	01	this is the address we are currently writing to in the code space of the device
Byte4	Register Start Address Lo	23	in this case we want to write to register address 0x0123
Byte5	Quantity of Registers to write Hi	00	We will be writing a variable amount of bytes at a time
Byte6	Quantity of Registers to write LOW	10	in this case we want to write to 10H or 16 registers
Byte7	Byte Count	20	If byte count is the same as number of Registers, dealing with 8 bits. If byte count is the same as number of Registers, dealing with 16 bits.

Byte #	8 bits	Byte #	16 bits
Byte8	Data 1	Byte8	Data1 Hi
Byte9	Data 2	Byte9	Data1 Lo
Byte10	Data 3	Byte10	Data2 Hi
Byte11	Data 4	Byte11	Data2 Lo
[…]		[…]	
Byte22	Data 15	Byte38	Data16 Hi
Byte23	Data 16	Byte39	Data16 Lo
Byte 24	Error Check HI	Byte40	Error Check HI
Byte 25	Error Check LO	Byte41	Error Check LO

[Notice]: Byte 7 is used as a byte count. Thus if the byte count is the same as the number of registers to write then we know we are dealing with 1 byte registers. Similarly, if the byte count is double the number of registers, we are dealing with 2 byte registers.

The slave device with ID=11 will answer the master within a few milliseconds with the following response.

Byte #	Field Name (Hex)	Data	Description
Byte1	Slave Address	11	destination node ID
Byte2	Function	10	this is a multiple write command
Byte3	Register Start Address Hi	00	starting address we are writing to, hi byte
Byte4	Register Start Address Lo	01	start address low byte
Byte5	Quantity of Registers Hi	00	Number of registers to be written to, hi byte
Byte6	Quantity of Registers Lo	0A	Number of registers, low byte
Byte7	Error Check (CRC) HI byte	XX	The CRC is calculated using the CRC
Byte8	Error Check (CRC) LO byte	XX	routine described previously

Example of the Multiple-Write Command
The Master sends the Multiple-Write querie:

Slave Address	Function	Starting Address Hi	Starting Address Lo	Quantity. of Regs Hi	Quantity. of Regs Lo	Byte Count
11	(10h) 16	(01h) 1	(23h) 35	0	3	6

Data 1 Hi	Data 1 Lo	Data 2 Hi	Data 2 Lo	Data 3 Hi	Data 3 Lo	CRC Hi Byte	CRC Lo Byte
(00h) 00	(0Ah) 10	(00h) 00	(0Bh) 12	(00h) 00	(0Ch) 13	xx	xx

Slave Address	Function	Starting Address Hi	Starting Address Lo	Quantity. of Regs Hi	Quantity. of Regs Lo	CRC Hi Byte	CRC Lo Byte
11	10	(01h) 1	(23h) 35	0	3	xx	xx

2.3 CRC Error Correcting Details

The following is a collection of code snippets to get your application started:

static unsigned char auchCRCHi[ ] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
} ;

/* Table of CRC values for low–order byte */

static unsigned char auchCRCLo[ ] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
0x40
} ;

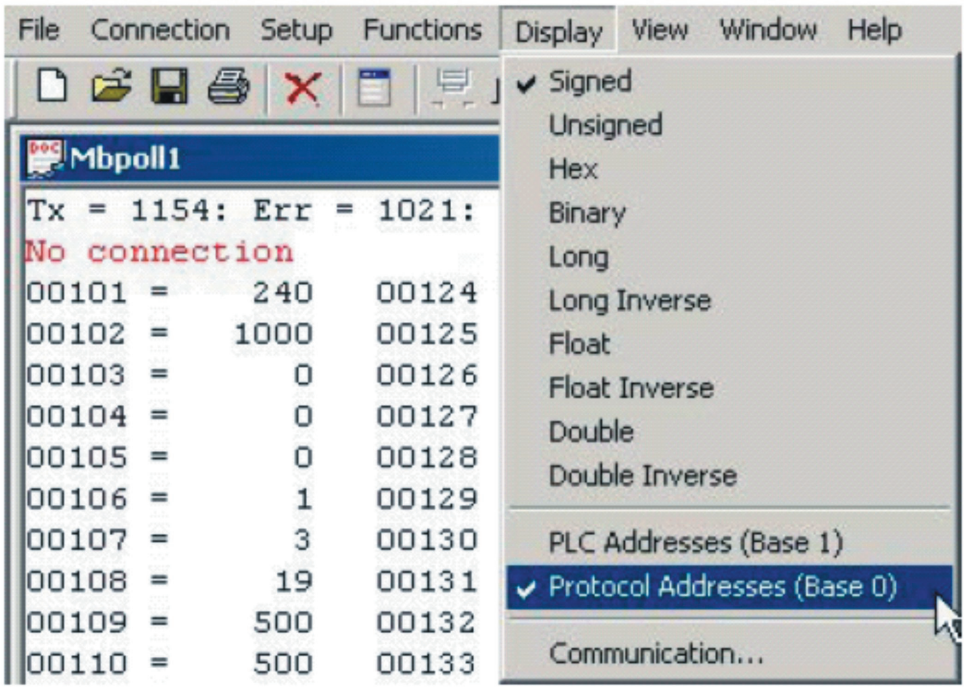
For example: to calculate the crc of the data in the message stored in memory location *puchMsgunsigned short CRC16 (unsigned char *puchMsg, unsigned char usDataLen

```
{
    unsigned char uchCRCHi = 0xFF ; /* high byte of CRC initialized */
    unsigned char uchCRCLo = 0xFF ; /* low byte of CRC initialized */
    unsigned uIndex ; /* will index into CRC lookup table */
    while (usDataLen--) /* pass through message buffer */
{
        uIndex = uchCRCHi ^ *puchMsg++ ; /* calculate the CRC */
        uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;
        uchCRCLo = auchCRCLo[uIndex] ;
    }
    return (uchCRCHi << 8 | uchCRCLo) ;
}
```

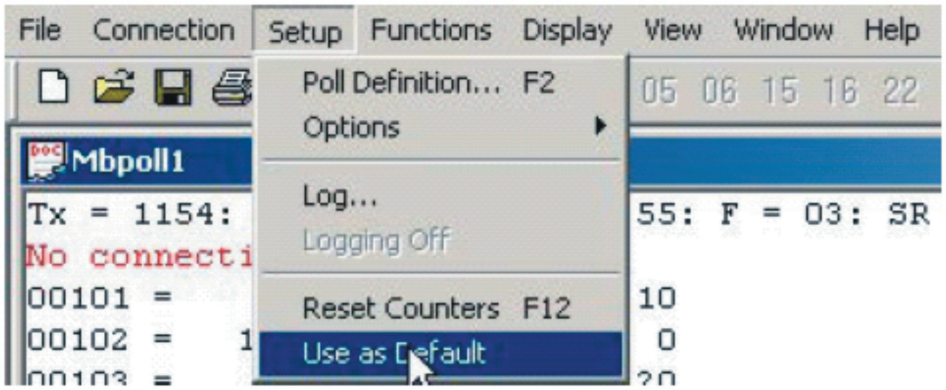
2.4 Modbus Poll Software

Modbus Poll is a simple modbus communications tool developed by Witte Communications http://www.modbustools.com/modbus_poll.asp that can be used to read and write registers of modbus devices. The following is a brief set of instructions for communicating with a device.

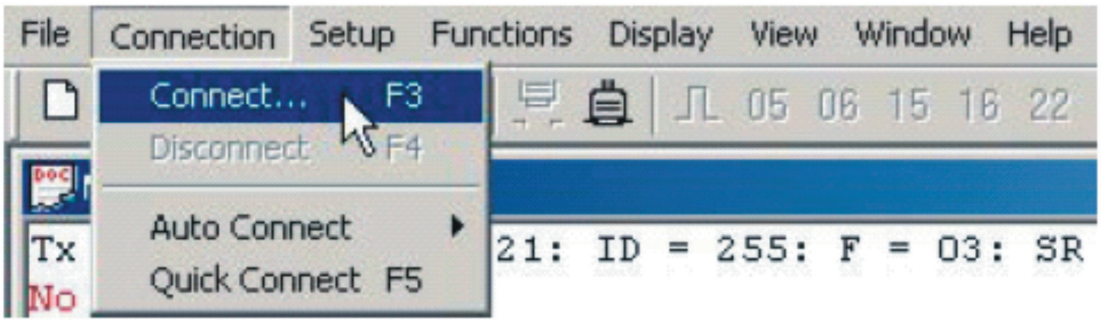
The first time Modbus Poll is used, it should be set to base 0 addressing. This is done by selecting “Protocol Addressing (Base 0) “ from the Display menu:



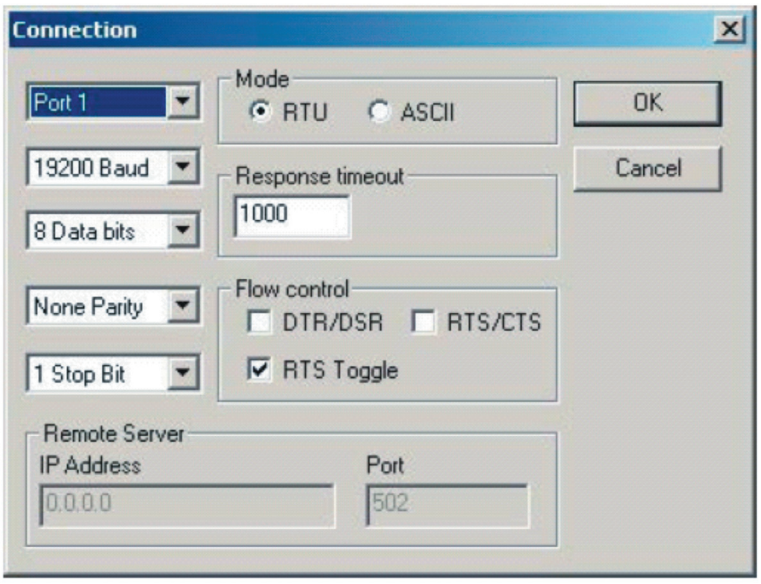
the Setup menu:



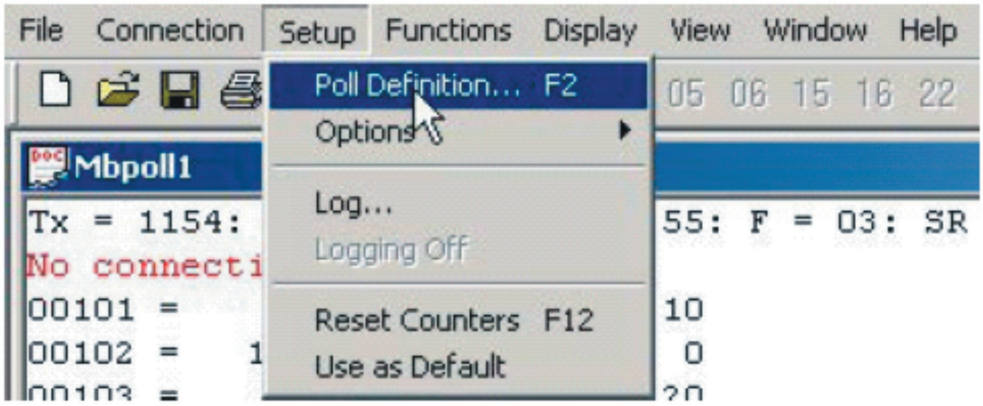
At this point, the connection to the device needs to be established. Select “Connect...” from the Connection menu:



Unless the device has specifically been setup for 9600 baud, the default connections settings should be as follows:



After the connection is established, it is necessary to setup the poll definitions. This is done by selecting "Poll Definition..." from the Setup menu:



Within the Poll Definitions dialog window, there are several parameters that need to be set.

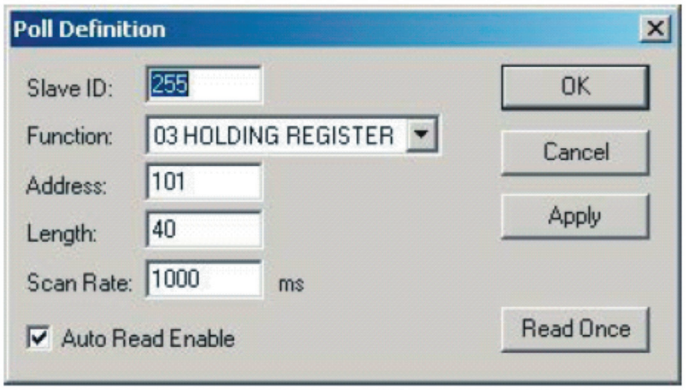
Slave ID is the modbus address of the device being read or written. (255 is the generic address to which all devices will respond.)

Function should be set as 03 HOLDING REGISTER.

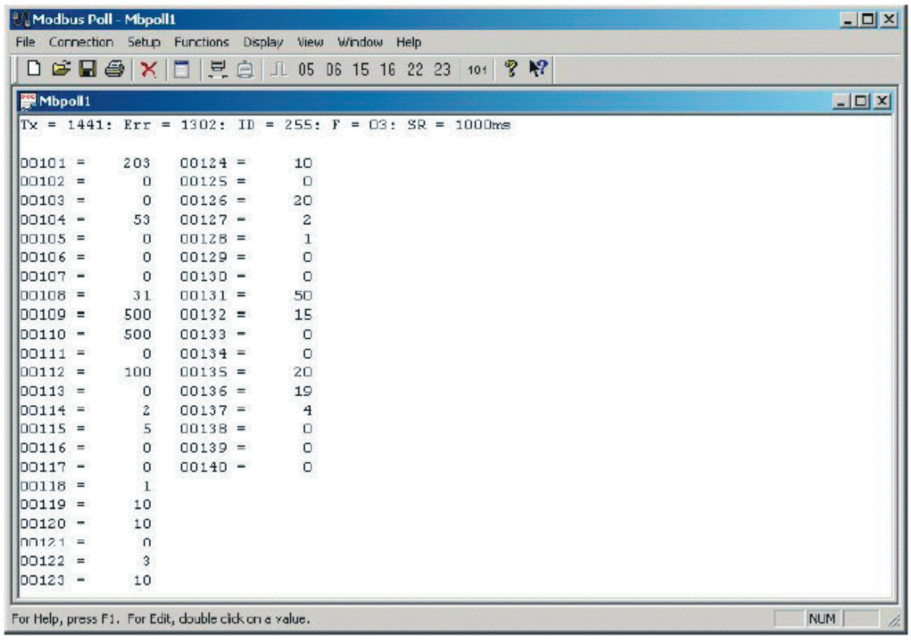
Address is the starting address of the registers to be read.

Length is the number of registers to be read.

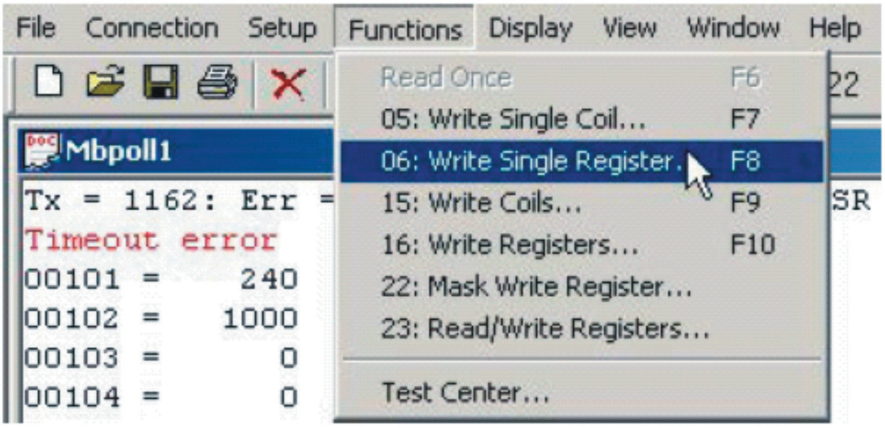
Scan Rate is the frequency with which the device will be polled.



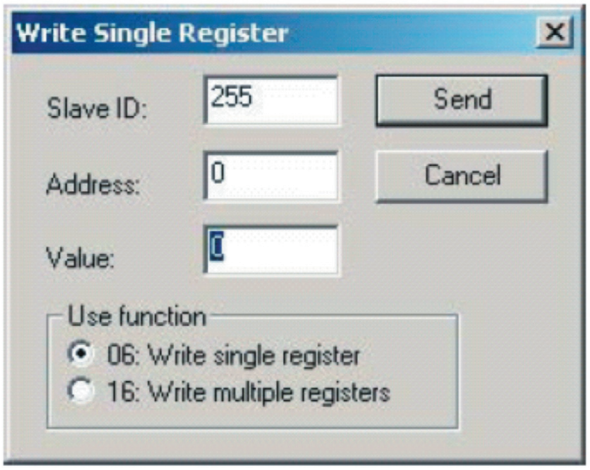
Once the Poll Definitions have been setup and applied, the main window will show a list of each register address and its corresponding value.



In order to write a value to a specific register, select “06 Write Single Register...” from the Functions menu:



Slave ID is the modbus address of the device. Address is the address of the register that will be written. Value is the value being written.



3 Instructions for Updating Devices with Temco ISP

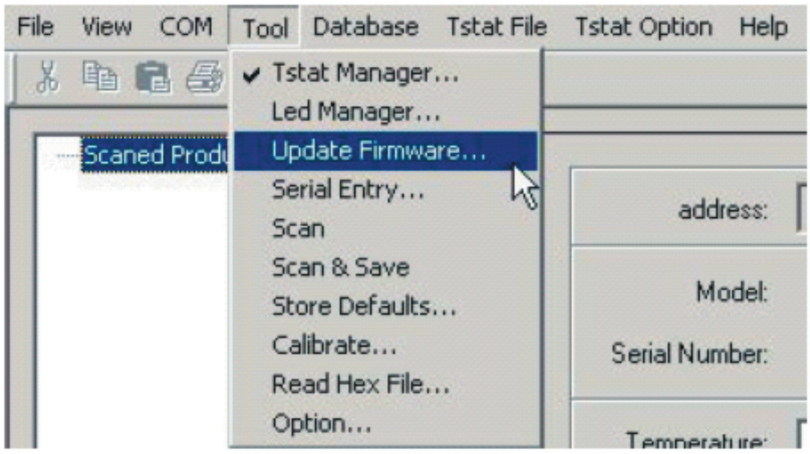
For TEMCO devices that utilize the Temco ISP, the flash update must be done using the provided NWT3000. To perform a firmware update, follow these instructions:

1.) Download and install the NWT3000 software: http://www.temcocontrols.com/ftp/software/9TstatSoftware.zip

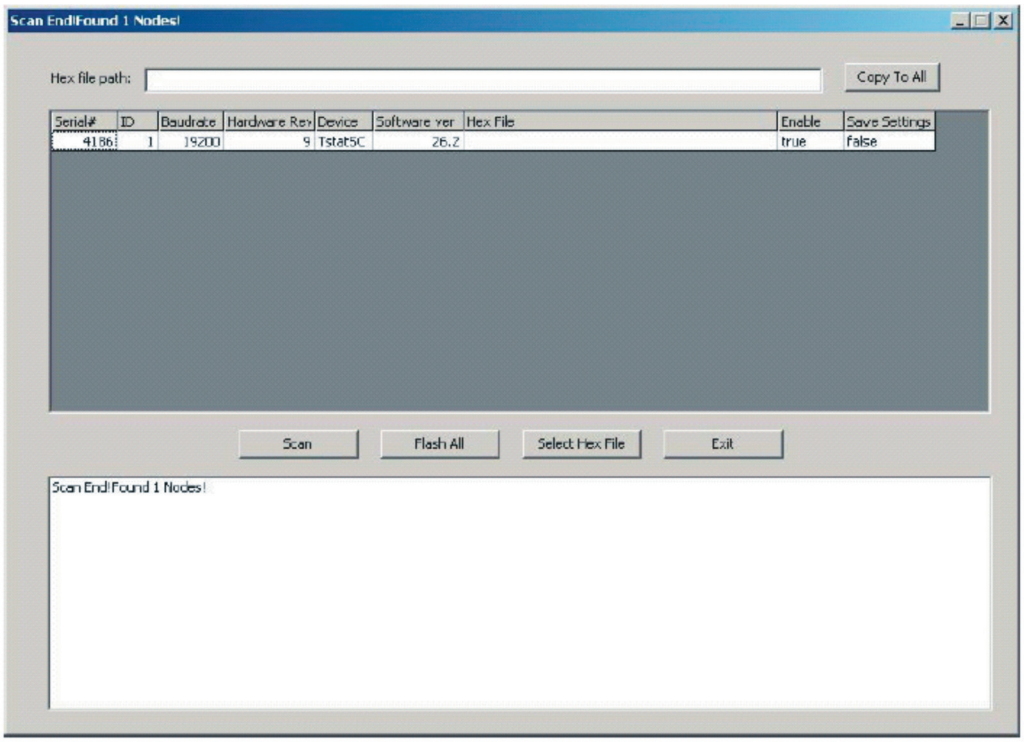
2.) Connect the device to the serial port of your computer using the RS232-485 converter included in the package.

3.) Power up the device.

4.) Open the NWT3000 software and select Update Firmware from the Tool menu:



5.) The software will now open the Update Firmware window and will scan for available devices.



6.) For each device that is found, you can specify the hex file to be used for the update. Do this by clicking in the Hex File column of the row you wish to specify. Alternatively you can click Select Hex File and then Copy to All if all devices are to receive the same file. You can also choose to save the current settings or to load the default settings by selecting True or False from the Save Settings column.

7.) At this point simply click Flash All and the software will update each device one by one.

3.1 Protocol for Developers Wanting to Update Devices with Temco ISP

All devices programmed with Temco ISP are capable of being updated over the RS485 network. The master on the network sends a command to a particular device, which forces it to go into a 'flash update mode'. The device first resets itself and then jumps to the 'In System Programming' (ISP) code section. Note that all non-volatile parameters should be read and saved prior to this for safe keeping.

NOTE: Multiple-Write Command of the Modbus protocol is used.

3.1.1 Protocol

In order for the front end to communicate with the ISP flash, a series of registers have been defined, which are used as control registers for the Update functions. Reading and writing to these registers will allow the Front end to monitor the status of the update process. They are stored in the non-volatile memory space to keep track of the steps attempted and completed. Below is a description of these control status registers.

Register	Register address	Description
EEPROM_VERSION_NUMBER	4	Software Version
EEP_ADDRESS	6	ID number of the device
EEP_UPDATE_STATUS	16	Update Register state

Table 1. Flash Update Function Registers

It is important to note 'EEP_UPDATE_STATUS' which is located at register address 16. Writing to this register will cause the device to either reset itself, erase its flash or start programming depending on the action being taken. Below is a description of the values and explanation of the EEP_UPDATE_STATUS register.

Function	Value	Description of EEP_Update_Status
Update initialize	7Fh	Tell the Tstat to reset and jump into the ISP to be in update mode
Update ready		Tstat is in the ISP and ready to update
Erase flash	3Fh	Tell the Tstat to erase Flash Memory
Erase done		Erase Flash Memory done
Start Programming	1Fh	Start Programming - In upload state
Normal State	01h	Update is complete, tstat reboots with new flash image

Table 2. EEP_UPDATE_STATUS register value description

- For the device to jump into update mode, a write command of value 7Fh must be sent to the EEP_UPDATE_STATUS. The device will then reset itself and run in ISP mode. Note: the device will not send any response in this step. To verify the T3module is in ISP mode, the same write command must be sent again (write 7Fh to register #16), at which point the T3module will respond with a regular modbus response. This is necessary for clearing the Interrupt vectors and making sure all RAM memory is cleared.

- All Modbus communication commands are always followed by a response. This Flash Update Protocol makes use of that criteria and thus only sends a response once the action has been completed. Therefore the 'update initialize' and 'erase flash' step require a longer timeout period than the 'programming' step. (250ms and 500ms respectively)

- Sending a write command of value 3Fh to EEP_UPDATE_STATUS will force the device to erase its entire flash memory. Once the response is received, the device is ready to download the data of the new firmware.

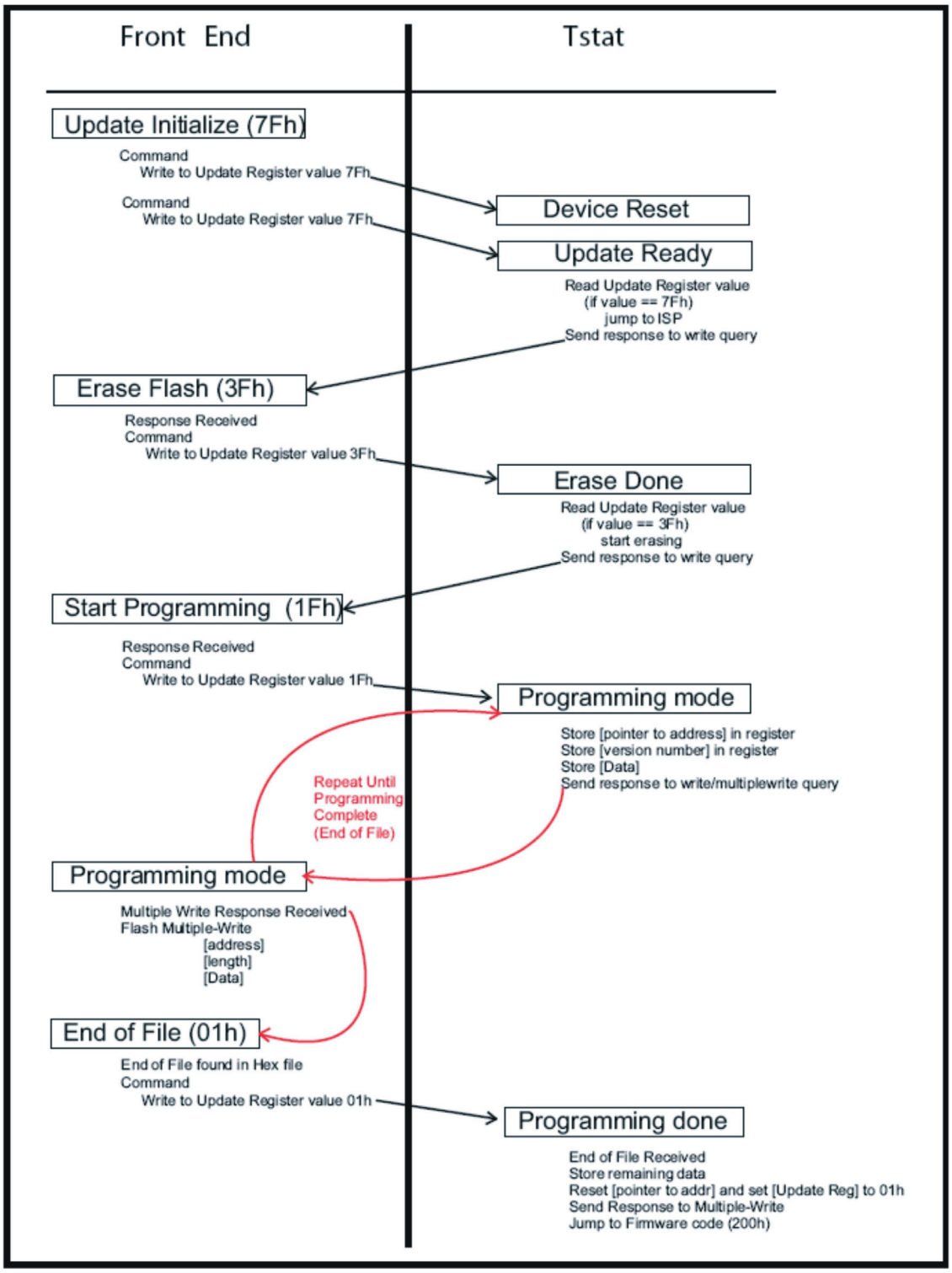
- Sending a write command of value 1Fh to EEP_UPDATE_STATUS will let the device know it is about to receive new firmware. The device is now ready to accept the new hex file and will maintain a running tally of the current programming location in the EEP_UPDATE_PTR.

- At this point, the data must be sent using the multiple-write command. Packets can be of size 1 data byte to a maximum of 128 data bytes.

- In the event of an interrupted flash update, the master can poll the EEP_UPDATE_PTR and begin programming from this location.

3.1.2 Example of a Programming Routine

The ISP has been designed using polling vectors rather than Interrupt vectors in order to free up as many interrupts for the program itself. Given that polling is now used, communications is more susceptible to timing and response delay problems. Therefore, when sending a write function or multiple-write function to the ISP device, a short timeout delay is required before receiving a response (≈20ms). If a response was not received during that period of time the FRONT END would need to resend the data once again. Below is a diagram representation of the Flash-Update Protocol.



3.1.3 Example of a Programming Routine (Front End Side)

UPDATE INITIALISE

1 - Send Modbus Write Command to address Update_Register value 7Fh
The device will reset itself. Make sure all volatile infomation be saved prior to this step
Device will not send a respond

2 - Send Modbus Write Command to address Update_Register value 7Fh again
A response will be received if the Device has properly reset itself and booted under ISP mode

ERASE FLASH

3 - Send Modbus Write Command to address Update_Register value 3Fh
A response will be received once the Device has properly Erase all Flash Memory
This will step require a longer response timeout period (approx 500ms)

TART PROGRAMMING

4 - Send Modbus Write Command to address Update_Register value 1Fh
A response will be received once the Device has properly set itself for programming mode

PROGRAMMING MODE

5 - Extracting data from Intel Hex file. A typical line would look like the following:
:10 0080 00 AF5F67F0 602703E0 322CFA92 007780C3 FD

6 - Verify checksum
10 + 00 +08 + 00 + AF + ... + C3 + FD = 900
If two last digits of the sum is zero, Hex file is correct

7 - Send data using Modbus Multiple-Write Command
Address 0080h
Data length of 10h
Data AF5F67F0 602703E0 322CFA92 007780C3

8 - Repeat step 5 through 7 until end of Hex file is reached
IMPORTANT NOTE to ensure proper reset of the device, the value at address register 0000h of the Goal chip must remain as FF.
Most (but not all) of Temco's Hex file will contain this line:
:03 0000 00 020200 F9
Data written to the Goal Flash register MUST be modified from 020200 to FF0200

END OF FILE

9 - End of file found in Hex file
:00 0000 01 FF
Bit 7 and 8 are 01

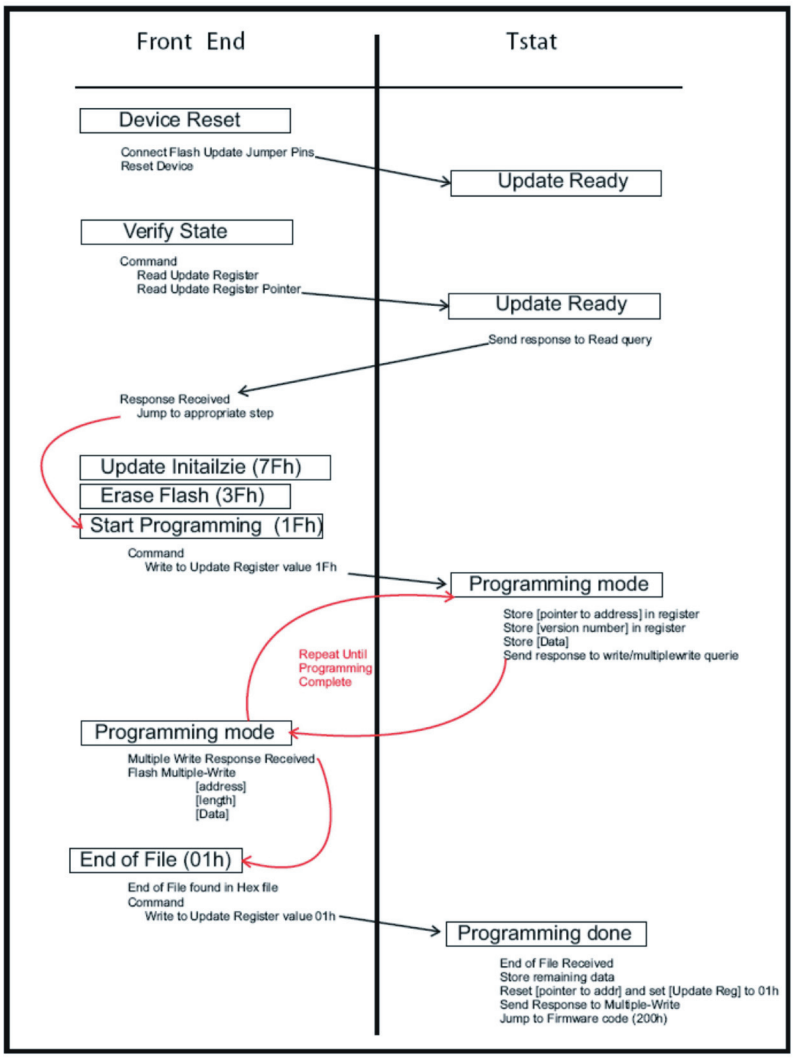
10 - Send Modbus Write Command to address Update_Register value 01h
This will cause the device to reset itself and boot in normal operation mode

3.1.4 To Resume a Previously Interrupted Programming Routine

The EEP_UPDATE_STATUS register keeps track of which step is being performed during the update protocol and the EEP_UPDATE_PTR keeps track of which register is currently being written to.

- If the device was in the Erase Flash mode, the EEP_UPDATE_STATUS register will read 3Fh. The Front End is then required to repeat this step and follow up from there.
- If the device was in the Programming mode, the EEP_UPDATE_STATUS register will read 1Fh. The Front Ends then needs to read the EEP_ UPDATE_PTR. Thus, in order to resume this step the Front End needs to re-write to this register again and then follow up from there.

The following diagram represents the update resume procedure.



IMPORTANT:

In order for the device to jump into the ISP mode, it has to reset itself. Upon reset, if the value at address register 0000h is FF the device will jump to the ISP code section. This is a hardware criteria of the Goal Chip and an efficient way to jump to In System Programming mode while clearing all buffers. The front end must ensure that only value FF is to be written to address register 0000h. When reading the hex file, there will be a line such as this:

Data of the new firmare — Modified data to be uploaded

;03 0000 00 020200 F9 -------- need to change to this to -------- ;03 0000 00 FF0200 FC

(Intel Hex format described below):

3.1.5 Intel Hex File

All firmware files produced by our compilers are saved under the Intel Hex file format. This format of record can be broken down in its different fields as described below.

3.1.5.1 Example of an Intel Hex file

Take for instance a typical message such as the following:

```
:ll  aaaa  tt  D1D2D3D4  D5D6D7D8  D9D0D1D2  D3D4D5D6  ee
:10 0080 00 AF5F67F0  602703E0  322CFA92  007780C3  61
```

- The first character (:) indicates the start of a record.
- The next two characters indicate the record length (10h).
- The next four characters give the load address (0080h).
- The next two characters indicate the record type. (00)
- Then we have our data
- The last two characters are a checksum (sum of all bytes + checksum = 00).

Record types:

- 00 - Data record
- 01 - End of file record
- 02 - Extended segment address record
- 03 - Start segment address record
- 04 - Extended linear address record
- 05 - Start linear address record

3.1.6 Intel Hex File

In the case where the device is locked, there is still a possibility to reboot the device and upload a new firmware. This requires to physically link the jumpers of the Flash Update Jumper pins during restart:

- Power down the device
- Link the jumpers of the Flash Update Jumpers
- Power up the device

Doing the above steps will force the device to be in ISP mode so that new firmware can be loaded. In order to return to normal operation once the upload has been done the Jumper needs to be removed and power need to be recycled.