

## **Ethernet Snap I/O For 21Plus!**

## Contents

<b>1</b>	<b>INTRODUCTION .....</b>	<b>3</b>
<b>2</b>	<b>SPECIFICATIONS.....</b>	<b>3</b>
<b>3</b>	<b>MODBUS/TCP SUPPORT .....</b>	<b>4</b>
<b>3.1</b>	<b>The Modbus Register Model and Data Tables .....</b>	<b>5</b>
<b>4</b>	<b>CONFIGURATION (TUNING) .....</b>	<b>6</b>
<b>4.1</b>	<b>Module Types .....</b>	<b>6</b>
<b>4.2</b>	<b>Modbus/TCP Tuning.....</b>	<b>7</b>
4.2.1	Device Tuning.....	8
4.2.2	Digital Input Tuning .....	8
4.2.3	Analog Output Tuning .....	8
4.2.4	Digital Output Tuning.....	9
4.2.5	Analog Input Tuning.....	9
4.2.6	Special Tuning .....	9
<b>4.3</b>	<b>GCT Tuning .....</b>	<b>10</b>
4.3.1	Digital Outputs.....	11
4.3.2	Digital Output Pulse for Control:.....	12
4.3.3	Analog Outputs .....	12
4.3.4	Digital Inputs .....	14
4.3.5	Analog Inputs.....	14
<b>5</b>	<b>OPTO-22 'SIMPLE I/O' SETUP (SNAP-ENET-S64) .....</b>	<b>14</b>
<b>5.1</b>	<b>General.....</b>	<b>14</b>
<b>5.2</b>	<b>SNAP Simple I/O Physical Specification .....</b>	<b>15</b>
<b>5.3</b>	<b>The Front Panel.....</b>	<b>16</b>
<b>5.4</b>	<b>Setting the IP Address .....</b>	<b>16</b>
5.4.1	Procedure for setting IP Address .....	16
5.4.2	Reset to Factory Default .....	19
<b>6</b>	<b>TROUBLESHOOTING .....</b>	<b>19</b>
<b>6.1</b>	<b>Manually setting an output .....</b>	<b>21</b>
<b>7</b>	<b>AN EXAMPLE OF TUNING .....</b>	<b>21</b>
<b>8</b>	<b>WATCHDOG CONFIGURATION ( <i>IMPORTANT</i> ) .....</b>	<b>24</b>
<b>9</b>	<b>PROCESS CONTROL EVENT VARIABLES.....</b>	<b>28</b>
<b>10</b>	<b>SPECIAL CONFIGURATIONS .....</b>	<b>29</b>
<b>10.1</b>	<b>Counters.....</b>	<b>29</b>
10.1.1	GCT and tuning.....	29

## 1 Introduction

Modbus/TCP has been added to the System 21Plus!™ product as an alternative to using the standard serial interface to SNAP I/O modules. By using the Ethernet I/O interface to communicate with SNAP I/O modules, the modules now may be placed anywhere the network wiring travels. This allows the remote I/O to be placed where the signal is and as a result, exceed the limit of approximately 50 feet for straight serial connections. In addition, analog I/O may be read and written at a much higher data rate. Each System 21Plus!™ Scanner iBox™ supports up to three Modbus/TCP devices such as the OPTO-22 'Simple I/O' (SNAP-ENET-S64) unit. Each OPTO-22 'Simple I/O' supports up to sixteen (16) modules on a single rack with up to 64 I/O points total.



Figure 1 - OPTO-22 'Simple I/O'

The interface used to communicate with the remote I/O device is Modbus/TCP. This is an open protocol developed in 1999 with the mission to move MODBUS® protocols into the 21st century. As part of the name hints, standard and widely used TCP/IP, the common transport protocol of the Internet, provides the reliable data transport mechanism to move the Modbus protocol packets from machine to machine.

## 2 Specifications

The Ethernet I/O specifications are being listed separately for the OPTO-22 'Simple I/O' device and EGS Gauging's general support for Modbus/TCP. If a feature is listed in the OPTO-22 'Simple I/O' and not the Modbus/TCP, then this indicates that this is possible due to the OPTO-22 device rather than a standard capability of Modbus/TCP protocol.

Feature	OPTO-22 'Simple I/O'	Modbus/TCP
Poll rate (1)	30 ms	30 ms
Digital Inputs (Maximum)	64	64
DI Latch detection (On or Off) (2)	Yes	No
DI signals may be inverted	Yes	Yes
Digital Outputs (Maximum)	64	64
DO pulsed outputs	Yes	Yes
DO pulsed output resolution (3)	6 ms	6 ms
DO signals may be inverted	Yes	Yes
Analog Inputs (Maximum)	32	64
AI are read as 32-bit floating point	Yes	Yes
Analog Outputs (Maximum)	32	64
AO are written as 32-bit floating point	Yes	Yes
Grouping of Inputs and Outputs (4)	Yes	Yes

Table 1 – Feature Listing

**Notes:**

1. *It really depends on the total number of all points. The more points there are, the slower it ends up being polled. Also, polling faster than the GCT rate doesn't provide any benefits and would just load the network down.*
2. *Detection of short On pulses (Off-On-Off) or Off pulses (On-Off-On) is handled by the SNAP I/O module and available on some modules down to the sub-millisecond range.*
3. *The worst case will be when all digital and analog inputs are due to be polled. This could take up to 6 milliseconds. Otherwise, the resolution (clock) is approximately one millisecond.*
4. *I/O points of a similar kind may be grouped and then read/written as a group in a single packet, thus saving both time and network loading. The data points are not required to be contiguous, but may have a mixture of other I/O data type intermixed.*

### 3 Modbus/TCP Support

Modbus is a master/slave protocol with half-duplex transmission. Modbus/TCP is an encapsulation of standard Modbus protocol in TCP/IP form. In more detail, the Modbus/TCP is a TCP/IP based variant of the Modbus RTU protocol and covers the use of Modbus messaging in either an 'Intranet' or 'Internet' environment. The protocol uses binary encoding of data (RTU) and TCP/IP's error detection mechanism for detection of transmission errors. Standard Modbus defines a set of data and control functions to perform data transfer, slave diagnostic and PLC program download. A subset of the standard set of data and control functions has been employed in the EGS implementation of Modbus/TCP to perform the I/O necessary to interface to the SNAP I/O modules. The System 21Plus!™ product performs as the master and polls up to three slave devices identified by an IP and slave address. The System 21Plus!™ product supports limited slave diagnostics, but no PLC program downloading. The following table lists the conformance class Modbus functions and those currently implemented by EGS Gauging.

Modbus Function Code	Current Terminology	Classic Terminology	Currently Implemented by EGS (See Notes)
<b>Conformance Class 0</b>			
3 (0x03 hex)	Read Multiple Registers	Read Holding Registers	No
16 (0x10 hex)	Write Multiple Registers	Preset Multiple Registers	Yes (1)
<b>Conformance Class 1</b>			
1 (0x01 hex)	Read Coils	Read Coil Status	No
2 (0x02 hex)	Read Input Discrete	Read Input Status	Yes (2)
4 (0x04 hex)	Read Input Registers	Read Input Registers	Yes (3)
5 (0x05 hex)	Write Coil	Force Single Coil	Yes (4)
6 (0x06 hex)	Write Single Register	Preset Single Register	No
7 (0x07 hex)	Read Exception Status	Read Exception Status	Yes (6)
<b>Conformance Class 2</b>			
15 (0x0F hex)	Force Multiple Coils	Force Multiple Coils	Yes (5)
22 (0x16 hex)	Mask Write Registers	Mask Write Registers	No
23 (0x17 hex)	Read/Write Registers	Read/Write Registers	No

Table 2 – Supported Modbus/TCP functions

## Notes:

1. Writing of Initial configuration of SNAP I/O modules along with writing one or many analog outputs as floating point values.
2. Reading of digital inputs and digital input latch registers on SNAP I/O modules.
3. Reading of analog input channels on SNAP I/O as floating point values.
4. Writing a single digital output.
5. Writing multiple digital outputs and resetting digital input latch registers on SNAP I/O modules.
6. Exceptions are automatically read upon detecting that an error has occurred.

### 3.1 The Modbus Register Model and Data Tables

The Modbus data functions are based on a register model. A register is the smallest addressable entry with Modbus. Classic Modbus defines only two elementary data types. These types are referred to as 'discrete' and 'register'. A discrete type represents a binary bit value and is typically used to address output coils and digital inputs of a PLC. A register type represents a 16-bit integer value. These elementary data types are transferred in big-endian byte order. It is very common to transfer 32-bit float values as pairs of consecutive 16-bit registers as done in the OPTO-22 SNAP I/O. *(Note: Currently, reading/writing of registers in both 16-bit and 32-bit integers is not implemented.)*

Table	Classic Terminology	Modicon® Register Table	Characteristics	EGS Implementation
Discrete outputs	Coils	0:00000	Single bit, alterable by an app. program, (Read/Write)	Digital Output
Discrete inputs	Inputs	1:00000	Single bit, provided by an I/O system, (Read Only)	Digital Input
Input registers	Input registers	3:00000	16-bit quantity, provided by an I/O system, (Read Only)	Analog Input is read as two adjacent registers and converted to IEEE float.
Output registers	Holding registers	4:00000	16-bit quantity, alterable by an app. program (Read/Write)	Analog Output is written as two adjacent registers, converted from an IEEE float.

Table 3 – Discrete and Registers

The Modbus protocol defines these areas very loosely. The distinction between inputs and outputs and bit-addressable data items does not imply any slave specific behavior. It is very common that slave devices implement all tables as overlapping memory areas as in the case with the SNAP I/O 'Simple I/O' unit.

## 4 Configuration (Tuning)

The new GCT Function Blocks (FB) associated with the Ethernet I/O are in the same FB as the existing serial SNAP I/O FBs. This is the Process FB and usually is named Process1() and found in the Frame FB. The new FBs are ModbusCtrl1(), ModbusTCP1(), ModbusTCP2() and ModbusTCP3(). The configuration for ModbusCtrl1 is just to enable the Modbus functionality and read the configuration. In addition to the .aif file, there is a new file .cfg, to configure the Modbus/TCP.

Unfortunately, the OPTO-22 'Simple I/O' module requires to be told what modules are present on the rack, unless this information is stored in its flash. Thus, the configuration now lists the modules for each slot position. Another difference from the serial interface is that the analogs points are now read/written to the modules as floating point numbers rather than DAC integer numbers as before. This is a requirement from the OPTO-22 'Simple I/O' module. This may be just a minor difference in the configuration as scaling of the analog I/O is still required.

The first part of the tuning will cover the configuration of the Modbus device(s). Afterwards, the GCT tuning will be covered.

### 4.1 Module Types

A number of module type are supported on the OPTO-22 'Simple I/O' device. Below are the ones that have currently been configured into the Modbus/TCP software.

Configuration Name For Tuning	Ch	Lower Range	Upper Range	Module type
<a href="#">Snap_DI</a>	4	-	-	Any Digital Input
<a href="#">Snap_DO</a>	4	-	-	Any digital Output
<a href="#">Snap_AI_MA_20_to_20</a>	2	-20.000	20.000	SNAP-AIMA... -20 to +20 mA
<a href="#">Snap_AI_RMSA_0_to_10</a>	2	0.000	10.000	SNAP-AIRMS.. 0 to 10 Amps RMS
<a href="#">Snap_AI_MV_50_to_50</a>	2	-50.000	50.000	SNAP-AITM2... -50 to +50 mV
<a href="#">Snap_AI_MV_15_to_15</a>	2	-15.000	15.000	SNAP-AITM... -150 to 150 mV
<a href="#">Snap_AI_V_10_to_10</a>	2	-10.000	10.000	SNAP-AIV... -10 to +10 V
<a href="#">Snap_AI_RMSV_0_to_250</a>	2	0.000	250.000	SNAP-AIVRMS... 0 to 250 V RMS
<a href="#">Snap_AI_TI_ICTD</a>	4	0.000	1000.00	SNAP-AICTD... Temp Input
<a href="#">Snap_AI_TM_0_to_40K</a>	4	0.000	40000.0	SNAP-AIR40K-4 Resistive thermistor
<a href="#">Snap_AI_RTD_100_OHM</a>	2	0.000	200.000	SNAP-AIRTD 3-wire Platinum
<a href="#">Snap_AI_FREQ_0_25K</a>	2	0.000	25000.0	SNAP-AIRATE.. 0 to 25000 Hz Input
<a href="#">Snap_AO_MA_4_to_20</a>	2	4.000	20.000	SNAP-AOA-23.. 4 to 20 mA Output
<a href="#">Snap_AO_VDC_0_to_10</a>	2	0.000	10.000	SNAP-AOV-25.. 0 to 10 V DC Output
<a href="#">Snap_AO_VDC_10_to_10</a>	2	-10.000	10.000	SNAP-AOV-27.. -10 to 10 V DC Output
<a href="#">Snap_AO_MA_0_to_20</a>	2	0.000	20.000	SNAP-AOA-28.. 0 to 20 mA Output

Table 4 – Supported SNAP I/O Tuning Names

## 4.2 Modbus/TCP Tuning

As mentioned previously, a new tuning file has been created to contain the configuration data. The file must have the same resource name as the iBox, but have a file extension of **.cfg**. A sample configuration file is show below.

```

'                                     Modbus/TCP Configuration
'
' <- The tick is a comment marker. SW ignores all to right of the tick.
'
'
' Modbus TCP  host      slave  pollrate  name
device tcp    10.2.23.44    1        50     "Simple I/O"
'
' List SNAP Modules.....
SNAP_MODULE  0  Snap_DI
SNAP_MODULE  1  Snap_AO_VDC_10_to_10
SNAP_MODULE  2  Snap_DO
SNAP_MODULE  3  Snap_AI_V_10_to_10
'
'
' Type      Module position  GCT_Table  Latched (DIs)  Notes.....
DI          0      1        0          ON_LATCH      ' Remote Offset
DI          0      2        1          ON_LATCH      ' Remote Scan
DI          0      3        2          ON_LATCH      ' Sheet Break
DI          0      4        3          ON_LATCH      ' Roll Cut
'
group_write
AO          1      1        0          -4.8      4.85      ' Screw Speed
AO          1      2        1          -4.8      4.85      ' Motor control
'
DO          2      1        0          -4.8      4.85      ' Pulsed out
DO          2      2        1          -4.8      4.85      ' Motor 12
DO          2      3        2          -4.8      4.85      ' Scrap light
DO          2      4        3          -4.8      4.85      ' Lunch light
'
AI          3      1        0          -4.0      5.0       ' Temp 1
AI          3      2        1          1.0      -1.0      ' No Limit Checking
'

```

Listing 1 – Sample Modbus/TCP Configuration file

## 4.2.1 Device Tuning

### device

For each SNAP or Modbus/TCP device, the type and IP address must be called out. This instruction marks the start of tuning for *each* device. The arguments required are:

1. tcp Right now, the only choice, as the other possibility is serial Modbus, which, has not been implemented.
2. IP Address The IP address of the OPTO-22 SNAP I/O (*host*) unit.
3. Slave Address All Modbus devices have a slave address from 1 to 255. This is usually set to 1 by default.
4. pollrate The period of time between starting of one polling cycle where all data points are read/written and the next polling cycle. A value of 50ms or greater is recommended.
5. "name" A name for the Modbus device up to 32 characters.
6. port *Optional* TCP port number. Typically, TCP port 502 is the default for Modbus/TCP. No need to include unless it's different from 502.

### SNAP MODULE

The SNAP modules as ordered on the rack are listed with the particular type called out. The first column in table 4 above lists the set of names to be placed here. (*Note: The SNAP modules must be listed after the **device** is called out and prior to listing any of the individual I/O points.*)

1. Module position Position on the rack where the module is located starting from position 0.
2. Module name The module name as listed in table 4.

## 4.2.2 Digital Input Tuning

### DI

Digital inputs are entered by giving the point's location and whether or not it is a latched input. There is new tuning for digitals discussed in section 10.

1. Module Position of the module on the rack, starting at position 0.
2. position The position of the digital on the module, starting at position 1. For digitals, there are usually only four channels.
3. GCT table The location of this point 'n' relative to the array of points in ModbusTCPn.DigIns[ n ] array.
4. ON\_LATCH Indicates that this point is to detect and latch any Off-to-On digital transitions. (Only one latch type may be configured for a point.)  
This option is required only if the duration of the ON state is less than the rate at which GCT is executed.
5. OFF\_LATCH Indicates that this point is to detect and latch any On-to-Off digital transitions. (Only one latch type may be configured for a point.)  
This option is required only if the duration of the OFF state is less than the rate at which GCT is executed.

## 4.2.3 Analog Output Tuning

### AO

Analog Outputs are entered by giving the point's location and whether or not the output values are to be clamped to a user specified range. By default, the range listed in table 4 is used as a clamp range to restrict the output to.

1. Module Position of the module on the rack, starting at position 0.
2. position The position of the analog channel on the module, starting at position 1. For analogs, there are usually only two channels.



- |                |  |
|----------------|--|
| 3. GCT table   | The location of this point 'n' relative to the array of points in ModbusTCPn.AnaOuts[ n ] array.   |
| 4. Lower clamp | Indicates that this point is to limit the analog output to always be greater than this value. Optionally, leaving this and the next entry blank results in the default value from the table 4 being entered.     |
| 5. Upper clamp | Indicates that this point is to limit the analog output to always being less than this value. Optionally, leaving this and the previous entry blank results in the default value from the table 4 being entered. |

#### 4.2.4 Digital Output Tuning

##### DO

Digital outputs are entered by giving the point's location. Pulsed or timed output are available to all digital outputs and are configured in the GCT tuning.

- |              |  |
|--------------|--|
| 1. Module    | Position of the module on the rack, starting at position 0.  |
| 2. position  | The position of the digital on the module, starting at position 1. For digitals, there are usually only four channels. |
| 3. GCT table | The location of this point 'n' relative to the array of points in ModbusTCPn.DigOuts[ n ] array.                       |

#### 4.2.5 Analog Input Tuning

##### AI

Analog Inputs are entered by giving the point's location and whether or not the output values are to be clamped to a user specified range. By default, the range listed in table 4 is used as a clamp range to restrict the input to.

- |                |   |
|----------------|---|
| 1. Module      | Position of the module on the rack, starting at position 0.   |
| 2. position    | The position of the analog channel on the module, starting at position 1. For analogs, there are usually only two channels.   |
| 3. GCT table   | The location of this point 'n' relative to the array of points in ModbusTCPn.Analns[ n ] array.   |
| 4. Lower clamp | Indicates that this point is to limit the analog input to always be greater than this value. Optionally, leaving this and the next entry blank results in the default value from the table 4 being entered.     |
| 5. Upper clamp | Indicates that this point is to limit the analog input to always being less than this value. Optionally, leaving this and the previous entry blank results in the default value from the table 4 being entered. |

#### 4.2.6 Special Tuning

##### group write

An optional instruction to force the output points of the same type that follow to be written out in a single packet. Normally, the default is that all output channels of a similar type are written out individually. However, there may be times when all must be written to update multiple digital or analog outputs at the same instance. Thus, it is possible to output up to 64 digital outputs or 32 analog outputs in a single write. The first point that follows this instruction sets the output point type that will be used. All points of a similar kind that follow in the configuration list up until either the point type changes or the *start\_group* instruction is encountered are included in the group write.

**start\_group**

An optional instruction to disable the grouping of output points declared after this line.

**new\_module**

This optional instruction handles the situation where a module that is not presently in the supported list may be added and used. This is intended for new analog input or output modules that were not available at the time this software was developed. This has only been tested on analogs with only two channels. The command line would typically be :

```
new_module 4 Snap_AO_VXC_10_to_10 A7 2 -10.025, 10.025
```

Where:

- |                    |  |
|--------------------|--|
| 1. Data Point Type | The I/O points are classified as <ul style="list-style-type: none"> <li>• Digital Input = 1</li> <li>• Digital Output = 2</li> <li>• Analog Input = 3</li> <li>• Analog Output = 4</li> </ul>                    |
| 2. name            | The name to be used to reference this module later on when the SNAP_MODULE instruction is used.  |
| 3. Snap ID         | This is a hex number supplied by OPTO-22 that identifies the module to the OPTO-22 SNAP 'Simple I/O' unit.   |
| 4. Channels        | The number of channels on the module.  |
| 5. Lower clamp     | Indicates that this point is to limit the analog output to always be greater than this value. Optionally, leaving this and the next entry blank results in the default value from the table 4 being entered.     |
| 6. Upper clamp     | Indicates that this point is to limit the analog output to always being less than this value. Optionally, leaving this and the previous entry blank results in the default value from the table 4 being entered. |

### 4.3 GCT Tuning

This section will discuss the GCT tuning associated with the Modbus/TCP interface. As in most GCT configuration, there is an 'AIF' file where the configuration is entered. As in the past, the entries for the Process I/O points are entered after the frame and sensor specific configuration tuning.

Three new Function Blocks (FB) have been created and are located in the Process1() FB.

There is the overall controller FB named ModbusCtrl1. This FB is responsible for reading in the configuration file and starting the process threads that will perform the network I/O with the Ethernet I/O devices. The FB is controlled by one variable that must be enabled to activate the Ethernet I/O interface. This entry is shown below:

```
Frame1.Process1.ModbusCtrl1.Enabled := TRUE; # Overall control enable for
                                         # Modbus/TCP
```

The second new FB is named ModbusTCP, however, there are three instances of this FB number 1, 2 and 3. Similarly, for each Modbus device, there is an enable flag to turn-on that ModbusTCP FB. Only those FBs that are to be used should have their enable flag set to TRUE.

```

Frame1.Process1.ModbusTCP1.Enabled:= TRUE;      # 1st Modbus/TCP device is
                                                # active
Frame1.Process1.ModbusTCP2.Enabled:= TRUE;      # 2nd Modbus/TCP device is
                                                # active
Frame1.Process1.ModbusTCP3.Enabled:= TRUE;      # 3rd Modbus/TCP device is
                                                # active

```

The third new FB is named **ModbusDO**, and there are three instances of this FB number 1, 2 and 3. This FB handles just the digital output type channels. The output of these modules goes directly into the **ModbusTCP** FB. These FBs get their enable flag from the associated **ModbusTCP** FB.

### 4.3.1 Digital Outputs

The Digital Output data enters the **Frame1.Process1.ModbusDO** FB in either of two ways. The output state can be set locally or be read remotely via a VAR REFERENCE. An array controlling this is the Boolean **DO\_ByRef** array and may support 64 digital output channels. If a channel is set to TRUE, then the digital state will be read via a VAR REFERENCE, otherwise, another local array has the current channel state.

<b>DO_ByRef</b>	- A BOOLEAN array for each DO channel. Set to TRUE to use <b>DORef_X</b> var ref. Set to FALSE to use BOOLEAN contained in <b>DOWrite</b> array.
<b>DOWrite</b>	- A BOOLEAN array for each DO channel. Setting a channel in this array will turn ON/OFF the associated Digital Output channel.
<b>DORef_X</b>	- Var Reference to BOOLEAN state located elsewhere.
<b>Tm_Ref_0</b>	- Var Reference to DINT for time in milliseconds located elsewhere.
<b>DigOutTimes</b>	- A DINT array to set the pulse time (in ms) when NOT using a <b>DORef_X</b> var ref.

In the **Frame1.Process1.ModbusTCPm** (m=1,2, or3) FB, the two arrays **DigOuts** and **DigOutTimes** are set in the above **ModbusDO** FB. Each channel is represented by a particular position in the **DigOuts** array. Each channel can be inverted logic if required by a corresponding Boolean array named **DigOutInv**. Setting a channel in this array to TRUE, results in that output being inverted prior to being sent out via the **DigOutData** array to the Modbus interface.

```

DigOutData[n] = DigOuts[n] .XOR. DigOutInv[n] ;

```

The following line shows how to configure the second digital output channel to be inverted logic. By default, this channel setting is disabled; therefore, it only has to be entered where the signal logic must be reversed.

```

Frame1.Process1.ModbusTCP1.DigOutInv[1] := TRUE ;

```

**Digital Output Pulse:** Digital outputs may be pulsed under the following conditions.

1. Set both the state and duration at the same time. For Example:  

```

Frame1.Process1.ModbusDO1.DOWrite[3]      := TRUE ; # ON State
Frame1.Process1.ModbusDO1.DigOutTimes[3]   := 1000 ; # Milliseconds

```
2. Once the FB is executed, return both to previous state the next time the FB is executed :  

```

Frame1.Process1.ModbusDO1.DOWrite[3]      := FALSE ; # ON State
Frame1.Process1.ModbusDO1.DigOutTimes[3]   := 0 ;      # Milliseconds

```

The smallest pulse is the larger value - 10 milliseconds or the specified pulse width of the module.

### 4.3.2 Digital Output Pulse for Control:

Digital outputs may be used to provide control output pulses. Usually, there are two outputs associated with each control – one for positive increments and another for negative increments. Up to four control output (pairs) are available to use in each Frame1.Process1.**ModbusDOx** (x=1,2 or3) function block. The required parameters that must be set are as follows:

For example, To connect ScrewSpeed Control to a pair of digital outputs channels 0 & 1, the following items are tuned. The X value may be 0, 1, 2 or 3.

Enab_Cntrl_X	- A BOOLEAN variable that enables a control pair.
TmRf_XPos	- A Var Reference that points to the UDINT variable elsewhere in GCT where the Positive pulse width is set. (See .VRF below.)
TmRf_XNeg	- A Var Reference that points to the UDINT variable elsewhere in GCT where the Negative pulse width is set. (See .VRF below.)
DO_Chnl_XP	- The digital output channel to be pulsed for positive outputs.
DO_Chnl_XN	- The digital output channel to be pulsed for negative outputs.

Note: The control action is performed only when changes are detected in the Var Reference values.

The **.cfg** file would have the following for the DO channels located in the first and second positions on module 3.

'Type	Module	position	GCT_Table	LoLimit	HiLimit	Notes.....
DO	3	1	0			' Control Increment
DO	3	2	1			' Control Decrement

The **.vrf** file has the following for the Var References to read the positive pulse width and negative pulse widths from the SSControl Function Block.:

```
# ===== Digital Outputs =====
# References for Pulsed outputs
# Control Increment DO
Frame1.Process1.ModbusDO1.TmRf_0Pos^ref      := 'MDControl1.SSControl.outPPls' ;
#
# Control Decrement DO
Frame1.Process1.ModbusDO1.TmRf_0Neg^ref      := 'MDControl1.SSControl.outNPls' ;
#
```

The **.aif** file has the following configuration for setting up the control pulses. The first line enables this feature (first of four) to monitor and direct the pulse widths values to the selected digital output channels. The next two lines are the pair of channels to direct the positive and negative pulses to.

```
# Control Outputs
Frame1.Process1.ModbusDO1.Enab_Cntrl_0      := TRUE ;
Frame1.Process1.ModbusDO1.DO_Chnl_0P       := 0 ;
Frame1.Process1.ModbusDO1.DO_Chnl_0N       := 1 ;
```

### 4.3.3 Analog Outputs

The Analog Output data values are 32-bit floating-point (REAL) numbers and up to 32 or 64 channels are supported using 2 or 4 channel AO modules, respectively. The Analog Output data enters the Frame1.Process1.**ModbusAOx** function block in three different ways being set either locally or be read remotely via a VAR REFERENCE. This function block basically selects the

**source** for the data. The data is retrieved and optionally adjusted depending on the selection of simple algorithms. The variable arrays are as follows:

AbsDev	- ( <i>Absolute</i> ) A BOOLEAN array for each AO channel. Set to TRUE to use Var Refs to get and output the data as described below or set to FALSE to use one of the other algorithms.
PctDev	- ( <i>Percentage</i> ) A BOOLEAN array for each AO channel. Set to TRUE to use Var Refs to get and output the data as described below or set to FALSE to use one of the other algorithms.
AOREf_X	- A Var Reference to a REAL variable located elsewhere.
DevTgtRef_X	- A Var Reference to a REAL variable located elsewhere. (This may be used as either an <i>optional</i> offset or in a percentage calculation.)
AnaOuts	- A REAL array for each AO channel. Setting a value in a channel in this array without either AbsDev or PctDev enabled will output to the associated Analog Output channel.

The selection of each algorithm for a channel (**X**) is shown below:

```

If AbsDev[X] is TRUE Then
  AnaOuts[X] := AORef_X - DevTgtRef_X ;
or
If PctDev[X] is TRUE and DevTgtRef_X > 0.0 Then
  AnaOuts[X] := (AORef_X - DevTgtRef_X) / DevTgtRef_X ;
or
Otherwise,
  AnaOuts[X] := AOValues[X] ;

```

These variables that control the analog output **scaling** algorithms are configured in the Frame1.Process1.ModbusAOM (m=1,2 or 3) function blocks. The AnaOuts array exits this FB and goes into the Frame1.Process1.**ModbusTCPm** (m=1,2 or 3) function block via AnaOuts array.

Each channel in ModbusTCPm is represented by a particular position in the **AnaOuts** array. Every channel is *scaled* by the algorithm:  $Y = a * (X + c) + b$ . Shown below is the same algorithm using the actual names of the GCT arrays. This algorithm provides offset corrections before and after a gain multiplier is applied to the data value. The **AOdata** array is then out via the Modbus interface.

```
AOdata[n] = ( AOGains[n] * ( AnaOuts[n] + AOoffsets2[n] ) ) + AOoffsets[n] ;
```

*It is important to understand that the scaling is necessary to convert (adjust) the value to an output voltage (or current) supported by that particular module. This is different from the original SNAP I/O Interface.*

The following line shows how to configure the third Analog output channel to scale a GCT value that ranges from 0.0 to 100.0 to output a voltage in the range from -10.0 to +10.0. (By default, gains are 1.00 and offsets are 0.0.)

```

Frame1.Process1.ModbusTCP1.AOoffsets2[2] := -50.00 ;      # Subtract 50 to change the
                                                         # to -50.0 to +50.0
Frame1.Process1.ModbusTCP1.AOGains[2]    := 0.20 ;        # Divide by 5.0
Frame1.Process1.ModbusTCP1.AOoffsets[2]  := 0.00 ;        # No action required

```

### 4.3.4 Digital Inputs

The Digital Input data leave the FB as a Boolean array of up to 64 points. Each channel is represented by a particular position in the **DigIns** array. Each channel can be inverted logic if required by a corresponding Boolean array named **DigInInv**. Setting a channel in this array to TRUE will results in that input being inverted prior to being available for access in GCT. Below is an ordered list that shows how the digital data is handled

```
dDigIns[n]      = DigIns[n];           # Save last DI state
DigInData[n]    = [Get new DI state from Modbus]; # Get new DI state
DigIns[n]       = DigInData[n] .XOR. DigInInv[n]; # Invert logic
DigInsCngd[n]   = DigIns[n] .XOR. dDigIns[n];    # Detect changed states
                                                    # since last time.
```

The following line shows how to configure the fourth digital input channel to be inverted logic. By default, this channel setting is disabled, therefore it only has to be entered where the signal logic must be reversed.

```
Frame1.Process1.ModbusTCP1.DigInInv[3] := TRUE ;
```

### 4.3.5 Analog Inputs

The Analog Input data leaves the FB as a 32-bit floating-point (REAL) number array of up to 64 points. Each channel is represented by a particular position in the **Analns** array. Each channel is scaled by the algorithm:  $Y = a * (X + c) + b$ . Shown below is the same algorithm using the actual names of the GCT arrays. This algorithm provides offset corrections before and after a gain multiplier is applied to the data value. The **Aldata** array is the data read most recently from the Modbus interface.

```
Aldata[n] = [Get new AI data from Modbus]; # Get new AI value
AnaIns[n] = ( AIGains[n] * ( Aldata[n] + AIOffsets2[n] ) ) + AIOffsets[n] ;
```

The following line shows how to configure the third Analog Input channel to scale a Modbus read value that ranges from -50.0 to +50.0 mV to a GCT variable in the range from 42.0 (°C) to +1820.0 (°C). (By default, gains are 1.00 and offsets are 0.0.)

```
Frame1.Process1.ModbusTCP1.AIOffsets2[2] := 50.00 ; # Add 50 to change the
                                                    # to 0.0 to +100.0
Frame1.Process1.ModbusTCP1.AIGains[2]    := 17.78 ; # Multiple by 17.78
Frame1.Process1.ModbusTCP1.AIOffsets[2]  := 42.00 ; # Final offset
```

## 5 OPTO-22 'Simple I/O' Setup (SNAP-ENET-S64)

### 5.1 General

The SNAP-ENET-S64 is one of three general types of Ethernet capable modules sold by OPTO-22. The others are the more advanced *SNAP Ethernet I/O* and *SNAP Ultimate I/O* families.

This section will present the physical specifications of the SNAP-ENET-S64 module along with installing the IP address into the device's flash memory. Also, should that IP address be lost, resetting the unit to factory default state is covered too.

## 5.2 SNAP Simple I/O Physical Specification

The table that follows lists some of the features and specifications of the SNAP Simple I/O device.

Feature / Specification	Notes
Mounting racks style	SNAP-M racks (or B-Series)
Number of modules per mounting rack (Note 1)	4, 8, 12 or 16
Module types and maximum modules allowed (with largest rack)	
Digital Input or Output Modules (Note 2)	16
Analog Input or Output Modules (Note 3)	16
Operating Temperature	0 °C to 70 °C
Network Interface	IEEE 802.3 network
	10Base-T, 100Base-TX
Maximum Ethernet segment length (Note 4)	100 Meters
Cable Interface	RJ-45
Simple Network Management Protocol (SNMP)	No
Security (Note 5)	Yes

Table 5 – SNAP-ENET-S64 Specifications

### Notes:

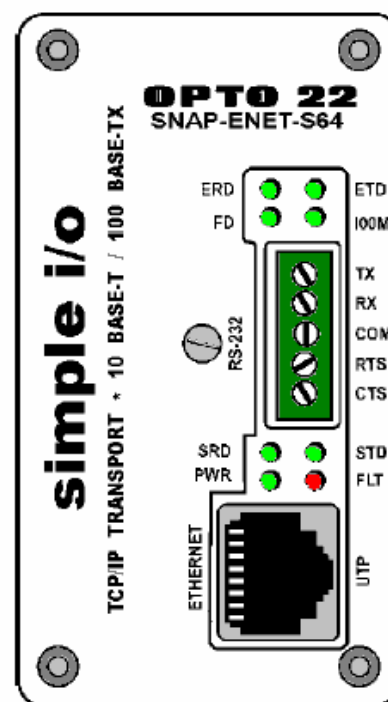
1. The SNAP-ENET-S64 may be mounted on SNAP-M16, SNAP-M32 or SNAP-M64 racks. (EGS Gauging has chosen the SNAP-M64 as the standard rack for all systems.)
2. Each digital module typically supports four channels. Modules may be placed in any position on the rack.
3. Each analog module typically supports two channels. Modules may be placed in any position on the rack.
4. OPTO-22 Documentation indicates length can be up to 100 meters with Category 5 or superior unshielded twisted pair (UTP). For 100 Mbps at this distance, use Category 5 or superior solid UTP. There is also a Category 5e (Enhanced) that has higher performance specifications.
5. Use OPTO-22 ioManager to limit access to Ethernet-based I/O units, either by allowing access only from specific computers or other devices on the network (IP filtering), or by limiting access to specific protocols, that are used with the I/O unit (port access). See the OPTO-22 ioManager User's Guide for more information on security.

## 5.3 The Front Panel

The following table describes the LEDs on the SNAP-ENET-S64 device.

LED	Description
ERD	Ethernet – Rcv Data
ETD	Ethernet – Tmx Data
FD	Full Duplex Mode
100M	Enet Link at 100 Mbps
SRD	Serial – Rcv Data
STD	Serial – Tmx Data
PWR	Power On
FLT	Microprocessor Fault

Figure 2 – LED Description



## 5.4 Setting the IP Address

All OPTO-22 SNAP 'Simple I/O' devices come from the factory without an IP address, thus before they can be used, this IP address must be written to flash. From OPTO-22, a utility program can be downloaded for performing this action.

### Program Description:

*ioManager* is configuration software for preparing SNAP Ethernet-based I/O and control systems (SNAP Ultimate I/O, SNAP Ethernet I/O, SNAP Simple I/O, SNAP-LCE, and SNAP-IT packaged systems) for use on Ethernet networks, including assigning IP addresses.

Available for download from OPTO-22 site:

<http://www.opto22.com/site/downloads/>

Scroll down and locate the "Configuration Software" for "ioManager".

Filename : **ioManager\_R71a\_000.exe**  
 Version : **R7.1**  
 Last Update : **2/2/2006**  
 Compatibility : **Windows 2000/ XP**  
 Size : **13,934 Kbytes**

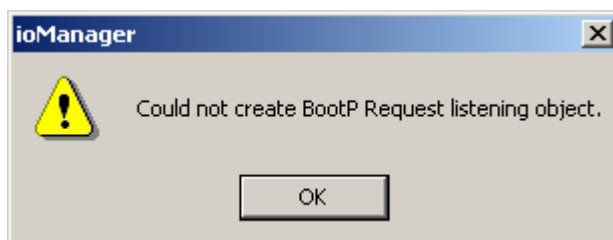
*This installer program is also available up on NAS2 in the folder named - \\NAS2\\Common\\Ethernet\\IO\\Opto22\_IOManager\\.*

### 5.4.1 Procedure for setting IP Address

1. If the IOManager has not been installed yet, that must be done before proceeding on any further.



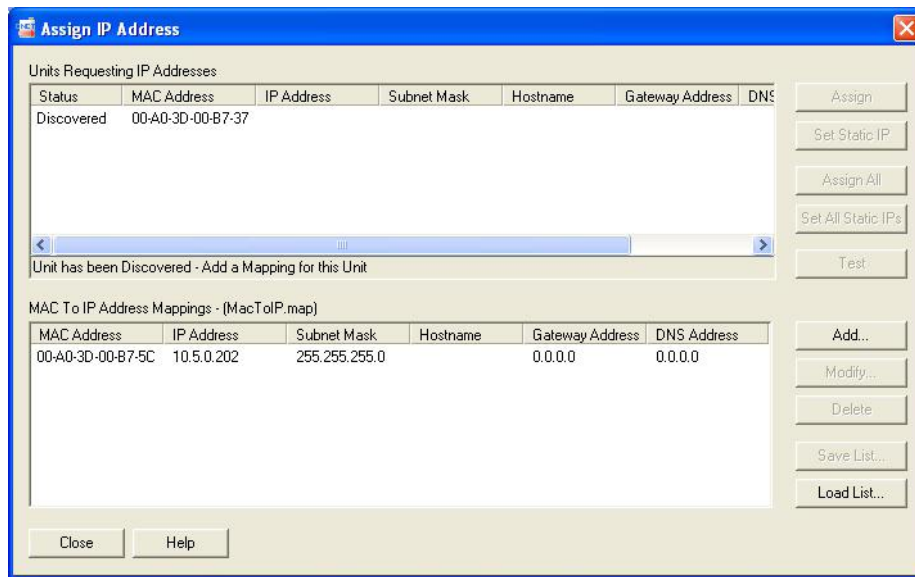
2. The OPTO-22 SNAP 'Simple I/O' device should be powered up and connected to the same network as the PC.
3. Start the Start > Opto22 > ioProject > ioManager to execute the ioManager program. If a small dialog pops up as shown below, then the bootpdNT service must be stopped before you can proceed.
  - Go to the Control Panel > Services program.
  - Click on the bootpdNT name in the list.
  - Now Right-click and select stop from the popup dialog.



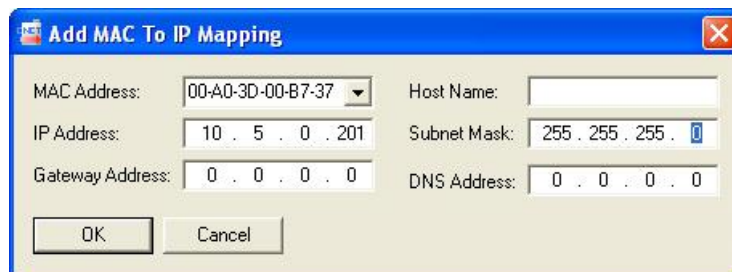
4. The ioManager window when it first starts.



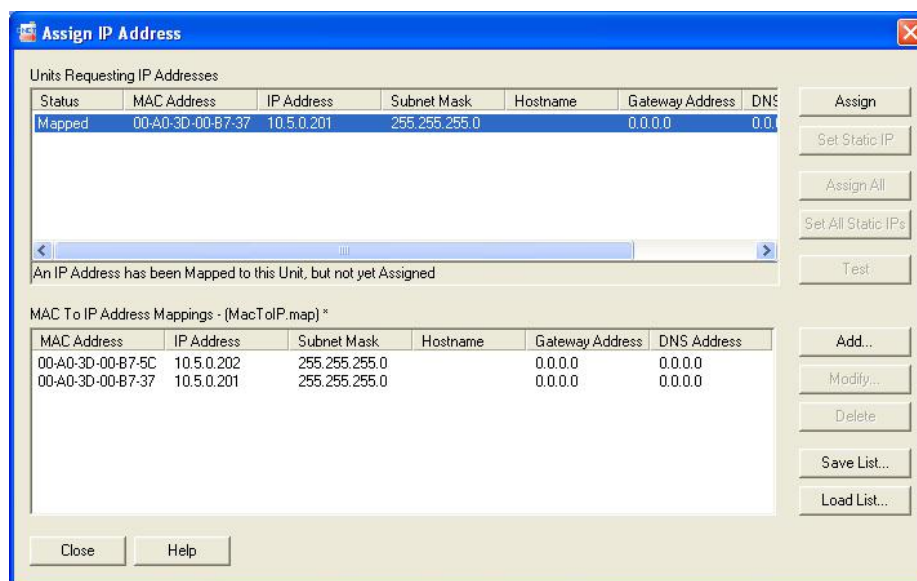
5. From the Menu, select Tools > Assign IP Address... You should now get the following display. If you don't see the MAC Address of the SNAP 'Simple I/O' device, then check all the network cables and possibly re-power the Snap I/O device. The MAC Address of each OPTO-22 device is printed on the top of the unit. It is always a 6 byte number shown in hexadecimal. The first three bytes should be 00-A0-3D.



- In the top half of the window, look for the exact MAC address. Double-click on the MAC Address for the device to be given the IP Address. Enter the IP Address that has been assigned this unit in the lower text box. Next, enter the Subnet Mask of 255.255.255.0 to the right of the IP Address text window. When both of these boxes have been filled in, the Send or Assign button should change from dimmed to normal. This signals that the information is ready to be sent to the specified MAC Address.



- Click the *Assign* button and a successful write to the unit will display the text **"Mapped"** or **"Static"** in the status column.



8. You can now exit the program. The IP address has been successfully written to flash in the OPTO-22 SNAP device.

### 5.4.2 Reset to Factory Default

Should the IP address be lost, the manual indicates that you reset the device to Factory Default state and repeat the previous method of setting the IP Address. To reset the device, follow the procedure.

1. Turn off power to the unit and remove the network cable.
2. Unscrew the four corner screws on the front of the unit.
3. Unscrew the center screw until it is loose and then pull the cover straight out with the center screw.
4. Inside the unit is a small PC board with a jumper. Carefully remove the jumper and place it on the other two wires.
5. Turn on power to the unit and count to ten. The green PWR LED should glow steadily at this time. This is the indication that the unit has been reset to factory default state. The IP Address and any other data in flash have now been erased. (The red LED may flash occasionally too.)
6. Remove the power to the unit again.
7. Carefully, move the jumper back to its original position.
8. Place the cover back on. The center screw must pass through a hole in the back of the unit as this screw holds the unit to the rack. Tighten with minimal force.
9. Screw in the four corner screws.
10. Reattach the network cable.
11. Follow procedure above on setting the IP address.

## 6 Troubleshooting

Should there be any problems with Modbus/TCP, two diagnostic tools are built in to the software. These are accessed by the (serial port COM1) console into the System 21Plus!™ Scanner iBox™. From the supervisor prompt, enter the resinst utility program.

## Modbus/TCP

```
resinst -e
```

Then go to the ModbusTCP1 FB by entering.

```
.Frame1.Process1.ModbusTCP1.
```

Now enter the following to display the Modbus/TCP configuration and status.

```
.prt_table := 1
```

An example print out is shown below. The current value for each channel is listed also.

```
[2.136.33] INTERNAL Frame1.Process1.ModbusTCP1.prt_table : BOOL = FALSE ;
[2.136.33] INTERNAL Frame1.Process1.ModbusTCP1.prt_table : BOOL = TRUE ;
Frame1.Process1.ModbusTCP1>
Modbus Configuration:
```

```
Modbus device # 0:.....
  Modbus/TCP   : Device IP: 10.2.23.44   Slave Addr: 1
  Pollrate    : 50
  Name        : 'Simple I/O'
  Pid         : 144   Conn Estab.   : 'OK!'
  Err Counter  : 1
  Last Error # : 0xa0000000
SNAP_Module____/_Slot / __Code____
Snap_DI                0    0x100
Snap_AO_VDC_10_to_10   1    0x0a7
Snap_DO                2    0x180
Snap_AI_V_10_to_10     3    0x00c
```

```
Digital Inputs_____
Group # 0:
Address: 1 Range: 4 Count: 4 GrpWrt: 0 Chngd: 0
  Addr_/_val_____/_GCT_Table_/_Idx_
    1    0                0    0    <on_latch>
    2    0                1    1
    3    0                2    2
    4    0                3    3
```

```
Digital Outputs_____
Group # 0:
Address: 9 Range: 4 Count: 4 GrpWrt: 0 Chngd: 0
  Addr_/_val_/_time_/_GCT_Table_/_Idx_
    9    1    0    0    0
   10    1    0    1    1
   11    0    0    2    2
   12    0    0    3    3
```

```
Analog Inputs_____
Group # 0:
Address: 25 Range: 2 Count: 2 GrpWrt: 0 Chngd: 0
  Addr_/_value_/_LoLimit/_HiLimit/_GCT_Table_/_Idx_
   25  -0.001  -4.000  5.000    0    0
   27  -1.666  { NO LIMIT RANGE } 1    1
```

```
Analog Outputs_____
Group # 0:
Address: 9 Range: 2 Count: 2 GrpWrt: 1 Chngd: 0
  Addr_/_value_/_LoLimit/_HiLimit/_GCT_Table_/_Idx_
    9    0.000  -4.800  4.850    0    0
   11   -3.300 -10.000 10.000    1    1
```

The second diagnostic is displayed by entering the following command.

```
.prt_errors := 1
```

An example print out is shown below. After the OPTO-22 device had been running, the network cable was removed and replaced. Later on the OPTO-22 unit was powered down and up again.

```
Frame1.Process1.ModbusTCP1>.prt_errors := 1
[2.136.34] INTERNAL Frame1.Process1.ModbusTCP1.prt_errors : BOOL = FALSE ;
[2.136.34] INTERNAL Frame1.Process1.ModbusTCP1.prt_errors : BOOL = TRUE ;
Frame1.Process1.ModbusTCP1>For Modbus/TCP#:      0
  Total errors :      10
  Last Error   : -1610612736
    Level 1 : modbus_poll
    Level 2 : write_modbus_digitals
    Level 3 : readDis_asints
Feb 24 23:45:11 2005 : 'modbus_openProtocol'      ' 0xa0000000
Feb 25 00:56:44 2005 : 'readAIs_asfloats'        ' 0x00000001
Feb 25 00:56:44 2005 : 'readDis_asints'          ' 0x00000001
Feb 25 00:56:45 2005 : 'modbus_openProtocol'      ' 0xa0000044
Feb 25 00:57:00 2005 : 'modbus_openProtocol'      ' 0xa0000000
Feb 25 01:13:44 2005 : 'readAIs_asfloats'        ' 0x00000001
Feb 25 01:13:44 2005 : 'readDis_asints'          ' 0x00000001
Feb 25 01:13:45 2005 : 'modbus_openProtocol'      ' 0xa0000044
Feb 25 01:13:45 2005 : 'write_single_DO'         ' 0x00000001
Feb 25 01:14:00 2005 : 'modbus_openProtocol'      ' 0xa0000000
```

When the Modbus/TCP interface fails to get a response from the OPTO-22 unit, it will first retry the request. If that fails, the TCP session to the device is closed and then a new TCP session is attempted with the unit. The Modbus/TCP interface will continue to retry to open the connection until it is successful.

## 6.1 Manually setting an output

Since your using the resinst tool, the outputs can be manually changed unless they are being written to from somewhere else in GCT. For example, set the 4<sup>th</sup> digital output (not a control output pulse type) to 1 on the first Modbus:

```
Frame1.Process1.ModbusDO1.DOWrite[3] := 1
```

Set the 2<sup>nd</sup> analog output to 2.0 is done as follows:

```
Frame1.Process1.ModbusTCP1.AnaOuts[1] := 2.0
```

*(Note: In the resinst tool, the channel number is entered **zero** relative, but the output is **one** relative.)*

## 7 An Example of Tuning

What follows are complete or portions of various tuning files used to interface an Ethernet SNAP I/O device and five modules into a System 21Plus system.

- DI for Remote Offsheet
- DI for Remote Scan
- DI for Remote Roll Cut
- DI for Remote ACP Hold (Non-standard)
- DI for Remote Sample Check button (using newly added feature)
- AI for ScrewSpeed voltage measurement
- DO for ScrewSpeed Control Positive Pulses
- DO for ScrewSpeed Control Negative Pulses

## Modbus/TCP

- DO for Product Alarm output.

```

'
'                               Modbus/TCP Configuration
'
'  <- The tick is a comment marker. SW ignores all to right of the tick.
'
'
'  Modbus TCP  host      slave  pollrate  name
device  tcp    10.5.0.201    1       50     "Simple I/O"
'
'  List SNAP Modules.....
SNAP_MODULE 0  Snap_DI
SNAP_MODULE 1  Snap_DI
SNAP_MODULE 2  Snap_AI_V_10_to_10
SNAP_MODULE 3  Snap_DO
SNAP_MODULE 4  Snap_DO
'
'                               Latched (DIs)
'Type      Module position  GCT_Table  LoLimit HiLimit  Notes.....
DI          0          1         0           ON_LATCH  ' Remote Offsheet
DI          0          2         1           ON_LATCH  ' Remote Scan
DI          0          3         2           ON_LATCH  ' Roll Cut
DI          0          4         3           ON_LATCH  ' Remote APC Hold Button
'
DI          1          1         4           ON_LATCH  ' Remote Sample Check
'
AI          2          1         0        -10.0   10.0    ' SSPD (0.0 to 10.0 Volts)
'
'group_write
DO          3          1         0           ' Control Increment
DO          3          2         1           ' Control Decrement
'
DO          4          1         2           ' Product Alarm'
'
'
'-----

```

Listing 1. .CFG file

```

#  Digital Inputs
Frame1.Process1.IncRoll           :=  FALSE ;    ##
Frame1.Process1.IncSplice         :=  FALSE ;    ##
Frame1.Process1.IncShtBreak       :=  FALSE ;    ##
Frame1.Process1.IncOffSheet       :=  TRUE  ;    ## <<<<
Frame1.Process1.IncScan           :=  TRUE  ;    ## <<<<
Frame1.Process1.IncEStop          :=  FALSE ;    ##
Frame1.Process1.IncScrap          :=  FALSE ;    ##
Frame1.Process1.IncSpare1         :=  TRUE  ;    ## For APC Hold
Frame1.Process1.IncSpare2         :=  FALSE ;    ##
Frame1.Process1.IncScrewSpd       :=  FALSE ;    ##
#
#

```

Listing 2a. .AIF – Standard Section from file

## Modbus/TCP

```
# =====  
# Modbus/TCP Ethernet I/O :  
# =====  
Frame1.Process1.ModbusCtrl1.Enabled := TRUE; # Overall control enable for Modbus/TCP  
Frame1.Process1.ModbusTCP1.Enabled := TRUE; # First Modbus/TCP device is active  
#  
Frame1.Process1.ModbusTCP1.DigInInv[0] := TRUE; # 1st DI input is inverted  
Frame1.Process1.ModbusTCP1.DigInInv[1] := TRUE; # 2nd DI input is inverted  
Frame1.Process1.ModbusTCP1.DigInInv[3] := TRUE; # 4th DI input is inverted  
#  
Frame1.Process1.IncScrewSpdRef := TRUE ; # Using var ref  
Frame1.Process1.IncRollRef := TRUE ; # Using var ref  
Frame1.Process1.IncSampChkRef := TRUE ; # Using var ref  
Frame1.Process1.IncSpare1Ref := TRUE ; # Using var ref (APC Hold)  
#  
# Control Outputs  
Frame1.Process1.ModbusDO1.Enab_Cntrl_0 := TRUE ; # Enable Control output  
Frame1.Process1.ModbusDO1.DO_Chn_0P := 0 ; # Output Chan# for Pos. pulses  
Frame1.Process1.ModbusDO1.DO_Chn_0N := 1 ; # Output Chan# for Neg. pulses  
#  
Frame1.Process1.ModbusDO1.DO_ByRef[2] := TRUE ; # Var Ref for Product Alarm  
#  
#  
#  
# =====  
#  
# TEMP for testing  
Frame1.Process1.ModbusTCP1.AIgains[0] := 1.0 ;  
Frame1.Process1.ModbusTCP1.AIoffsets[0] := 0.0 ;  
#
```

Listing 2b. .AIF – New for Modbus/TCP Section from file

```
#####
#           MODBUS ADDITIONS
#           ( Note: Do not use spaces inside tick (') marks.)
#####
# =====
# ===== Digital Inputs =====
# Remote Offsheet DI
Frame1.Process1.OffSheetDigInRef^ref := 'Frame1.Process1.ModbusTCP1.DigIns[0]' ;
Frame1.Process1.OffSheetDigInRef^scan := T#50ms ;
#
# Remote Scan DI
Frame1.Process1.ScanDigInRef^ref      := 'Frame1.Process1.ModbusTCP1.DigIns[1]' ;
Frame1.Process1.ScanDigInRef^scan    := T#50ms ;
#
# Roll Cut DI
Frame1.Process1.RollRef^ref           := 'Frame1.Process1.ModbusTCP1.DigIns[2]' ;
Frame1.Process1.RollRef^scan         := T#50ms ;
#
# Remote APC Hold DI
Frame1.Process1.Spare1Ref^ref         := 'Frame1.Process1.ModbusTCP1.DigIns[3]' ;
Frame1.Process1.Spare1Ref^scan       := T#50ms ;
#
# Remote Sample Check DI
Frame1.Process1.SampChkRef^ref        := 'Frame1.Process1.ModbusTCP1.DigIns[4]' ;
Frame1.Process1.SampChkRef^scan      := T#50ms ;
#
# =====
# ===== Analog Inputs =====
# SSPD AI
Frame1.Process1.ScrewSpdRef^ref       := 'Frame1.Process1.ModbusTCP1.AnaIns[0]' ;
Frame1.Process1.ScrewSpdRef^scan     := T#50ms ;
#
# =====
# ===== Digital Outputs =====
# References for Pulsed outputs
# Control Increment DO
Frame1.Process1.ModbusDO1.TmRf_0Pos^ref := 'MDControl1.SSControl.outPPls' ;
#
# Control Decrement DO
Frame1.Process1.ModbusDO1.TmRf_0Neg^ref := 'MDControl1.SSControl.outNPls' ;
#
# Product Alarm DO
Frame1.Process1.ModbusDO1.DOREf_2^ref  := 'Frame1.Process1.ProductAlarm' ;
#
#
```

Listing 3. .VRF – Portion showing Modbus/TCP Section from file

## 8 Watchdog Configuration ( *Important* )

The Simple I/O module should be configured to reset any Digital or Analog Output channels should the network connection to the iBox stop for any reason. The Opto22 ioManager program is used to perform this one time configuration.

1. Start the OPTO22 ioManager program from the Start menu. Select the Tools Menu and click on the menu item *Inspect*.



- Enter the IP address at the top if not correct and key refresh. Click the 'Status Write' button and enter a timeout value, possibly in the range, from 3000 to 10000 msec, for the Comm. Watchdog timeout. Once entered, click the 'APPLY' button and then refresh to verify the write. (Note: The 'APPLY' button must be clicked after each change entered.)

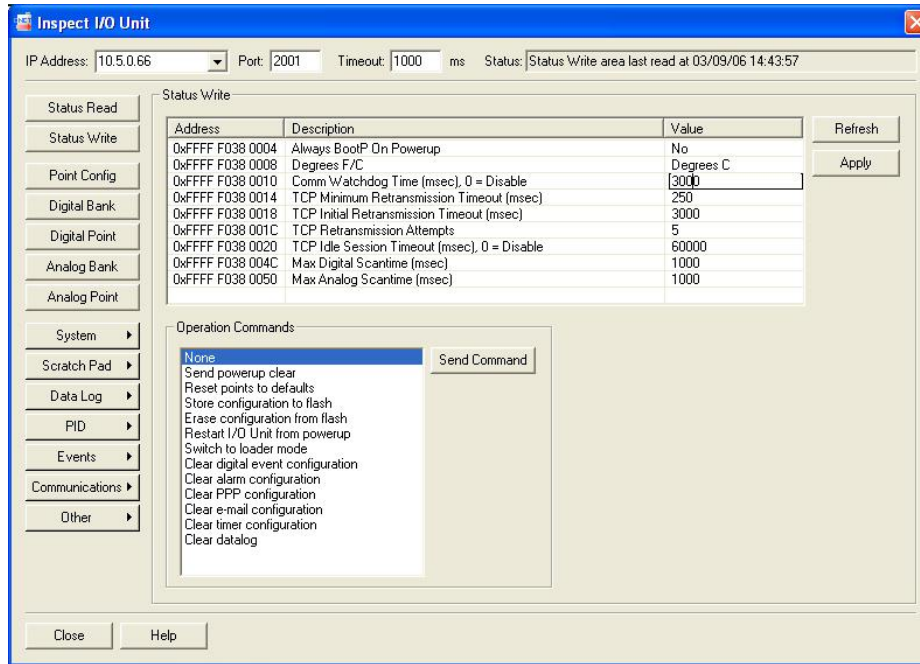


Figure 4. The Status Write Display from ioManager showing entry being set.

- Finally for each Digital Output that requires a reset (safe) value should communications be lost to the iBox, the following settings must be changed. Click the 'Point Config' button and find the module and channel on that module. The example below shows the fourth module (3) and first channel on that device (point 12). In the Value column, enter a 0 or 1 for the reset (safe) digital output state for the '**Watchdog Output Value**' line. Next set the '**Watchdog Enabled**' line to 'Enabled'. The example below has been set to turn off the digital output when the watchdog is triggered. Click the 'APPLY' button to write the settings to the device. Click 'refresh' to verify the write was successful.

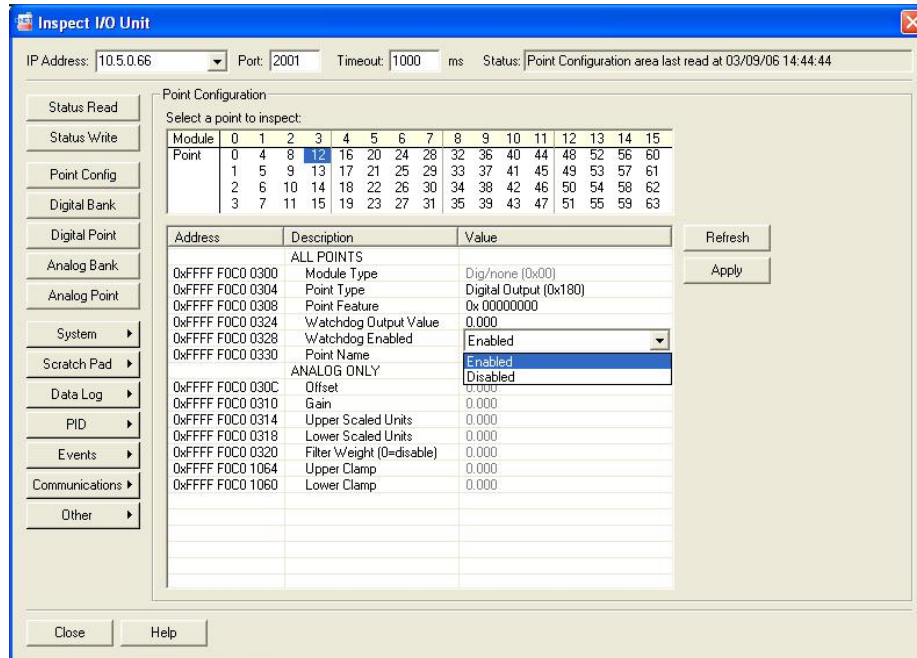


Figure 5. The Status Read Display from ioManager is shown upon start.

- Repeat Step 3 for each output that requires a reset state. Once all channels have been configured, proceed on to the last step.

- The final and *very important action* is to save the configuration changes to the flash internal to the SNAP-ENET-S64 device. Return to the first display by clicking the 'Status Write' button. The top list box is where the TCP timeout was entered. In the lower List box titled 'Operations Commands', click and highlight the 'Store configuration to flash' line option and click the 'Send Command'. Upon command completion, a message indicating the results is shown. Once done, the program may be closed.

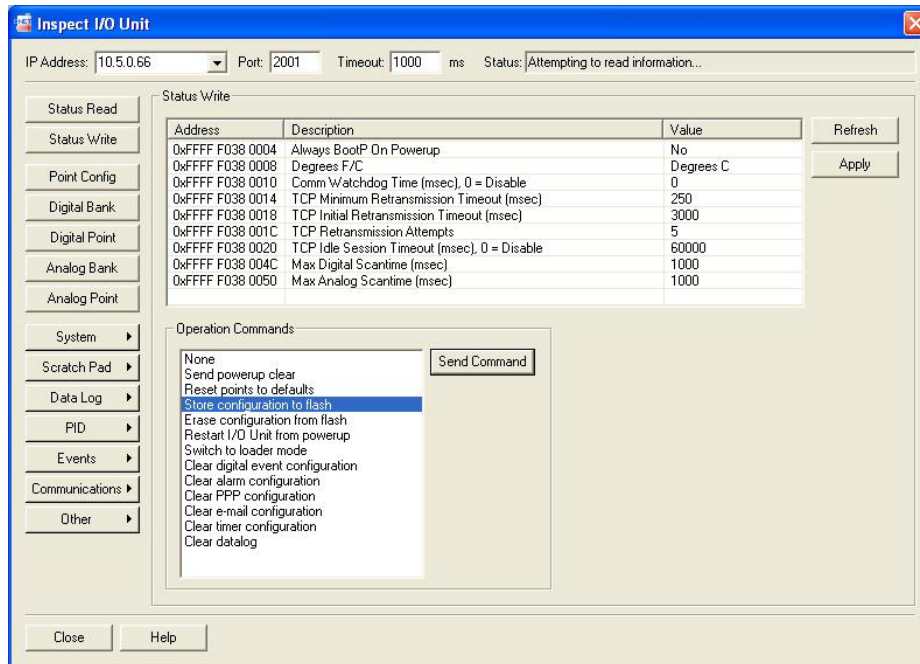


Figure 6. The Status Write Display saving configuration to Flash.

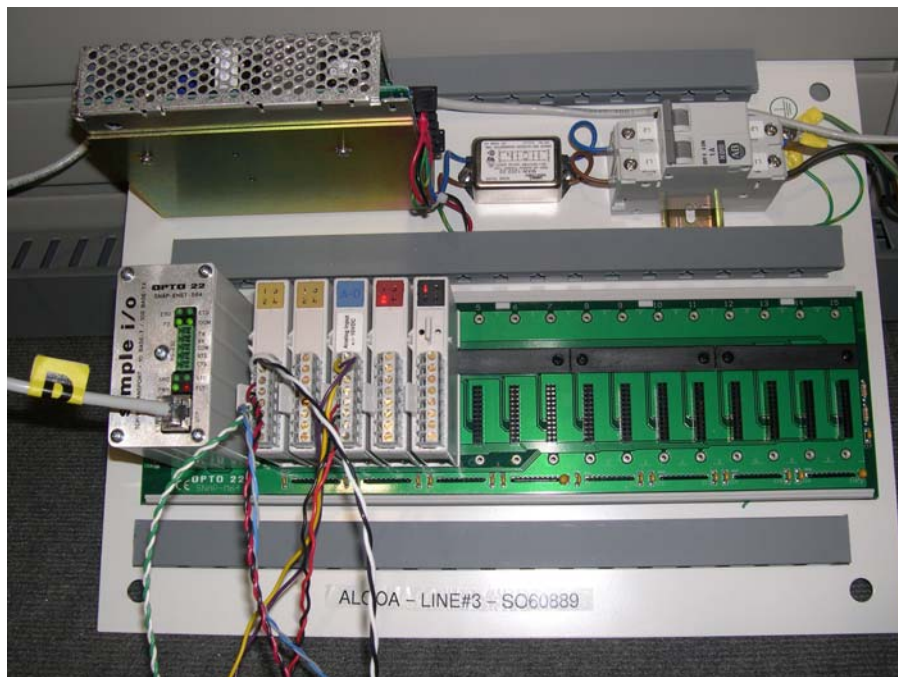


Figure 7. A SNAP-ENET-S64 .M64 Rack and modules.

## 9 Process Control Event Variables

System events such as Roll Cut or Remote Scan are initiated when I/O detects a signal on an assigned input channel. The following table lists events from the Process FB and the associated variables that are used. The ModbusTCP column are only suggested combinations to use, but the SNAP I/O direct connection column could be used for most situations. (*Obviously, for Roll Cut, the (hardwired) DigInShot1.Offshot would not work with ModbusTCP inputs.*)

Event	SNAP I/O via direct connection Frame1.Process1...	SNAP I/O via ModbusTCP Frame1.Process1...	Other Frame1.Process1...
Roll Cut	IncRoll (BOOL) DigInShot1.OffShot	IncRollRef (BOOL) RollRef (Var Ref to BOOL)	RollTest (BOOL)
E Stop	IncEStop (BOOL) EstopDigInRef (Var Ref to BOOL) (Inverted)	IncEStopRef (BOOL) EstopRef (Var Ref to BOOL)	
Sheet Break	IncShtBreak (BOOL) ShtBreakDigInRef (Var Ref to BOOL) (Inverted)	IncShtBreakRef (BOOL) ShtBreakRef (Var Ref to BOOL)	
OffSheet	IncOffSheet (BOOL) OffSheetDigInRef (Var Ref to BOOL) (Inverted)	IncOffSheetRef (BOOL) OffSheetRef (Var Ref to BOOL)	OffSheetTest (BOOL)
Splice	IncSplice (BOOL) OffSheetDigInRef (Var Ref to BOOL) (Inverted)	IncSpliceRef (BOOL) SpliceRef (Var Ref to BOOL)	
Scan	IncScan (BOOL) ScanDigInRef (Var Ref to BOOL) (Inverted)	IncScanRef (BOOL) ScanRef (Var Ref to BOOL)	ScanTest (BOOL)
Scrap	IncScrap (BOOL) ScrapDigInRef (Var Ref to BOOL) (Inverted)	IncScrapRef (BOOL) ScrapRef (Var Ref to BOOL)	ScrapTest (BOOL)
Air Wipe	IncAirWipe (BOOL) HomelnRef (Var Ref to BOOL) (Inverted)	IncAirWipe (BOOL) HomelnRef (Var Ref to BOOL) (Inverted)	
Scan Override	IncOverRide (BOOL) (OverRideDigInRef (Var Ref to BOOL) AND OverRideDigInRef2 (Var Ref to BOOL)) (Inverted)	IncOverRideRef (BOOL) OverRideRef (Var Ref to BOOL)	OverRideTest (BOOL)
Remote Sample Check	IncSampChk (BOOL) SampChkDigInRef (Var Ref to BOOL) (Inverted)	IncSampChkRef (BOOL) SampChkRef (Var Ref to BOOL) (Inverted)	SampChkTest (BOOL)
Screw Speeds (ScrewSpeed)	Either: 1) IncScrewSpdRef (BOOL) or IncScrwSpdRef (BOOL), ScrewSpdRef (Var Ref to REAL) 2) IncScrewSpd (BOOL), AnaIn1.ana0 (REAL)		
Screw Speeds (ScrewSpeed2)	IncScrewSpd2Ref (BOOL), ScrewSpd2Ref (Var Ref to REAL)		
Screw Speeds (ScrewSpeed3)	IncScrewSpd3Ref (BOOL), ScrewSpd3Ref (Var Ref to REAL)		
Screw Speeds (ScrewSpeed4)	IncScrewSpd4Ref (BOOL), ScrewSpd4Ref (Var Ref to REAL)		

Table 6. Standard Process Events and related variables

In addition to the standard list are eight non-reserved events that may be used as required. These are the Spare events and deal only with Boolean (digital) states.

Spare1	IncSpare1 (BOOL) Spare1DigInRef (Var Ref to BOOL) (Inverted)	IncSpare1Ref (BOOL) Spare1Ref (Var Ref to BOOL) (Inverted)	Spare1Test (BOOL)
Spare2	IncSpare2 (BOOL) Spare2DigInRef (Var Ref to BOOL) (Inverted)	IncSpare2Ref (BOOL) Spare2Ref (Var Ref to BOOL) (Inverted)	Spare2Test (BOOL)
Spare3	IncSpare3 (BOOL) Spare3DigInRef (Var Ref to BOOL) (Inverted)	IncSpare3Ref (BOOL) Spare3Ref (Var Ref to BOOL) (Inverted)	Spare3Test (BOOL)
Spare4	IncSpare4 (BOOL) Spare4DigInRef (Var Ref to BOOL) (Inverted)	IncSpare4Ref (BOOL) Spare4Ref (Var Ref to BOOL) (Inverted)	Spare4Test (BOOL)
Spare5	IncSpare5 (BOOL) Spare5DigInRef (Var Ref to BOOL) (Inverted)	IncSpare5Ref (BOOL) Spare5Ref (Var Ref to BOOL) (Inverted)	Spare5Test (BOOL)
Spare6	IncSpare6 (BOOL) Spare6DigInRef (Var Ref to BOOL) (Inverted)	IncSpare6Ref (BOOL) Spare6Ref (Var Ref to BOOL) (Inverted)	Spare6Test (BOOL)
Spare7	IncSpare7 (BOOL) Spare7DigInRef (Var Ref to BOOL) (Inverted)	IncSpare7Ref (BOOL) Spare7Ref (Var Ref to BOOL) (Inverted)	Spare7Test (BOOL)
Spare8	IncSpare8 (BOOL) Spare8DigInRef (Var Ref to BOOL) (Inverted)	IncSpare8Ref (BOOL) Spare8Ref (Var Ref to BOOL) (Inverted)	Spare8Test (BOOL)

Table 7. Additional Spare Process Events and related variables

## 10 Special Configurations

### 10.1 Counters

Digital Counters have been added to the Modbus/TCP list of features. This capability is not provided with the standard SNAP-ENET-S64 ('Simple I/O') brain modules, but with higher priced SNAP brains such as the SNAP-PAC-EB1. This Ethernet brain supports additional capabilities such as thermocouple linearization and PID loop control. The Modbus/TCP driver resident in the iBox has been modified to use the counting capability provided by this module. The additional tuning to implement counting is relatively simple. With this feature, there is now also new variables in the GCT modbusTCPx function blocks that are available.

Any digital input module may be used as a counter along with any or all channels on that module. The maximum frequency that the counter is capable of is related to the digital input module. Documentation on a particular module lists the turn-on and turn-off times, but maximum rates are under 20 kHz.

The SNAP 2-Axis Quadrature Position Input module (SNAP-IDC5Q) allows a snap brain to resolve two axes of rotating position information from Quadrature encoder devices. The modules outputs a pulse to the brain upon each change in quadrature state. The brain counts the pulses and keeps track of the direction.

#### 10.1.1 GCT and tuning

The new variables in the modbusTCPx FB consist of a 32bit signed integer array of 64 points. Each channel in the array **DigInsCounters** represents a possible counter channel on the rack.

The other array is a Boolean array of 64 channels that allow a counter to be reset or cleared to zero. This array is named **DigInCntrReset**. Since the counter is new and its uses are so far specialized for customer requirements, their initial use will be confined to custom software designs.

```
Frame1.Process1.ModbusTCP1. DigInCntrReset[0] := TRUE ; ## Clear counter to zero
```

There are only minor changes to the tuning required to make use of counters. Two new tuning words are available to be placed next to SNAP digital module configuration in the .cfg file. The first is **COUNTER** and is placed last on the line. Use this for counters on standard digital input modules. The second tuning word is **IDC5Q** and is use for the special quadrature module. An example is shown below. One important note about the SNAP-IDC5Q is that the second input channel on the module is in *position 3* and not 2.

```
'
'                                     Modbus/TCP Configuration
'
' <- The tick is a comment marker. SW ignores all to right of the tick.
'
'
' Modbus TCP host      slave  pollrate      name
device tcp 10.210.40.44  1      50          "PAC EB1 I/O"
'
' List SNAP Modules.....
SNAP_MODULE  0  Snap_DO
SNAP_MODULE  1  Snap_DI
SNAP_MODULE  2  Snap_DI
SNAP_MODULE  3  Snap_AI_V_10_to_10
'
'
' Type  Module position  GCT_Table  Latched  Notes.....
DO      0      1      0      LoLimit HiLimit
DO      0      2      1
DO      0      3      2
DO      0      4      3
'
DI      1      1      0      COUNTER  ' Scrap Counter
DI      1      2      1
DI      1      3      2      COUNTER  ' Roll cut input
DI      1      4      3
'
DI      2      1      4      IDC5Q      ' Quad rotation counter 1
DI      2      3      5      IDC5Q      ' Quad Counter 2, Second channel is 3 !!!
'
AI      3      1      0
AI      3      2      1
'
' LVDT Bottom operator Side
' LVDT Bottom Drive Side'
```

Listing 4 – Sample Modbus/TCP Configuration file for counters

The Modbus/TCP driver in the iBox will set up the SNAP-PAC-EB1 to use either the standard digital input or 2-Axis Quadrature Position Input module automatically. Therefore, no configuration is required on the SNAP I/O brain module by the project engineer other than setting the IP Address and watchdogs. There is also new OPTO-22 'PAC manager' PC software that comes with the SNAP-PAC-EB1 brain that replaces the previous 'I/O Manager' software.

One final note on the tuning is related to the internals of the new SNAP-PAC-EB1 brain. OPTO-22 documentation on Modbus/TCP now lists two different register locations for configuring and activating counters. Another tuning word, **mb\_feature\_2**, may be required at some point in the future when the new register address is required. This word may be placed after the **device** line of tuning parameters in the .cfg file.