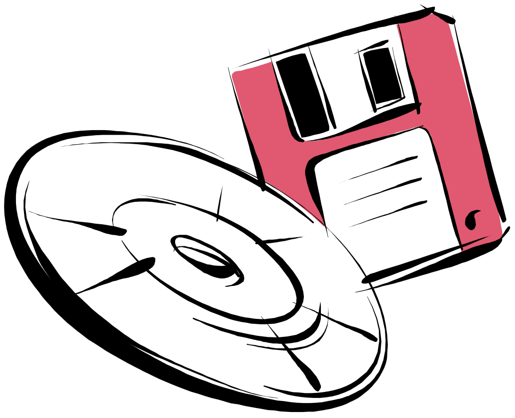
EtherNet/IP – Modbus XPort, NET232, and NET485



xxx-xxx-xxx

Document Version 1.08x

May 26, 2010

Grid Connect, Inc.
1630 W Diehl Rd
Naperville, IL 60563
(630) 245-1445

Revision List

Part Number: xxx-xxx-xxx

Rev	Date	Author	Comments
1.01	Apr 4, 05	JVK	Initial Release, Doc aligned with Firmware v1.08
1.02	May 4, 05	JVK	Update for Firmware v1.09, added binary file spec
1.03	June 29, 05	JVK	Update for Firmware v1.10
1.04	Mar 23, 06	JVK	Update for Firmware v1.11
1.05	July 1, 08	JVK	Update for Firmware v1.11d, fix type-o’s
1.06	Sept 24, 08	JVK	Update for Firmware v1.12
1.07	Dec 17, 09	JVK	Added hardware doc refs, hex values in tables
1.08		JVK	Update for Firmware v1.13

Table of Contents

1 Introduction

1.1 Purpose of This Manual

This document describes the operation of the Lantronix XPort customized by Grid Connect to enable Modbus ASCII/RTU devices with EtherNet/IP communications. The XPort is the primary component of the NET232 and NET485 products, therefore this manual applies to the operation of the EIP-MB versions of them as well.

NOTE: This manual is only a software/firmware reference. The following manuals should be consulted as hardware references for the XPort, NET232, and NET485. Only refer to the sections listed. Anything else in these manuals does not apply to the EIP-MB versions. Copies of these documents are included on the EIP-MB product CD.

- XPort_IntGuide.pdf - Section 2
- NET232_UM_800232_c.pdf - Sections 2.5, 2.7, 2.8, 2.9
- NET485_UM_800240_c.pdf - Sections 1.2, 1.3, 2.2, 2.3, 2.4

The EIP-MB firmware does not utilize everything available in the hardware of the above platforms. The differences are listed below:

- XPort - The EIP-MB firmware does not use the Configurable Pins.
- NET232 - The EIP-MB firmware does not use RTS and CTS.

1.2 Overview

The XPort behaves as an EtherNet/IP server device and a Modbus client capable of communicating with one Modbus slave. An XPort is mountable to a circuit board therefore it is convenient to add this component to an existing Modbus ASCII or RTU slave device.

The XPort is configurable via vendor specific objects in EtherNet/IP or via an upload of a configuration file. The user sets parameters in these objects to set up a table of Modbus function codes, coils, and registers supported by the slave. This table describes the mapping of coil and register data into the EtherNet/IP I/O messages. Internally, the mapped Modbus coils and registers are polled regularly by a Modbus master/client. The client continuously reads and writes internal buffers with the values of these coils and registers. This data is available via I/O data assemblies in the producer/consumer connection over EtherNet/IP.

1.3 Necessary Tools

The following tools are necessary for configuring this XPort:

- BOOTP Server – the XPort initially powers up and attempts to obtain an IP address using BOOTP. If no BOOTP server responds, the XPort will revert to a default IP address.
- EtherNet/IP Messaging Tool – can be a PLC or Software Tool, must be capable of establishing an explicit messaging connection and sending Get_Attribute_Single and Set_Attribute_Single EtherNet/IP explicit messages.
- TFTP Client Software – required to download/upload configuration files or to upload new firmware.

1.4 EtherNet/IP Terms of Usage

EtherNet/IP Technology is governed by the Open DeviceNet Vendor Association, Inc (ODVA). Any person or entity that makes and sells products that implement EtherNet/IP Technology must agree to the Terms of Usage Agreement issued by ODVA. See www.odva.org for details.

1.5 Documentation and Firmware Updates

Updates to this document and the XPort-EIP-MB firmware are available at:

http://developer.gridconnect.com

2 Features

The EtherNet/IP – Modbus XPort has the following features:

- Supports 1 EtherNet/IP I/O connection and 2 EtherNet/IP TCP connections.
- Mappings of Modbus Data to/from EtherNet/IP I/O Data are fully configurable.
- EtherNet/IP Module and Network Status LED states available to Modbus Slave.
- EtherNet/IP Identity Object attributes configurable.
- Ethernet MAC address configurable.
- Firmware is unlocked by default for ease of initial configuration, but it can be locked to prevent end-users from changing mappings or Identity Object attributes.
- Internal TFTP Server to upload new firmware and upload/download all of the configuration data – once one XPort is set up, download it’s configuration, then upload the file to the rest of your XPorts.
- BOOTP supported.

3 XPort Operation

3.1 Configuration Lock & Unlock

The XPort provides a means to unlock and lock the configuration, so it can or cannot be changed via Ethernet. Out of the box, the module powers up with the configuration unlocked since it does not yet contain a configuration. The configuration contains an attribute that can set the power-up default to be locked.

When the configuration is locked, the module behaves as follows:

- Attributes of Objects 0x64, 0x65, and 0x66 are read-only ("Set_Attribute_Single" service will fail).
- Object 0x6E (the Identity Configuration) does not exist.
- TFTP uploads of firmware and configuration files are denied.
- The Modbus client is fully functional.

When the configuration is unlocked, the module behaves as follows:

- Attributes of Objects 0x64, 0x65, 0x66, and 0x6E are configurable.
- TFTP uploads of firmware and configuration files are accepted.
- The Modbus client stops scanning (regardless of the values in the "Run/Idle Mode" and "Actively Scan Slave" attributes!!)

To toggle the lock, the following EtherNet/IP explicit message must be sent to the XPort:

Service Code	Class ID	Instance ID	Attribute ID	Data
0x45	0x67	0x89	0xAB	0xCD

Note that simply toggling the lock with the above message does not change the default lock state on power-up. The lock can be toggled any number of times. The lock will change to the default setting on the next power cycle.

Setting the default state of the lock at power-up is described in Section 3.2: Bridge Configuration Object (Class 0x64). When the configuration is locked, sending the above message is the only way to unlock it again.

3.2 Bridge Configuration Object (Class 0x64)

The various settings of the bridge and the Modbus port on the XPort can be configured through EtherNet/IP via Get_Attribute_Single (0x0E) and Set_Attribute_Single (0x10) Services directed at Vendor Specific Object Class 100 (0x64), Instance 1.

Attr ID (dec)	Attr ID (hex)	Access	Name	Data Type	Description
100	0x64	Get	Modbus Status	UINT	(See below for Bit Mask definition)
101	0x65	Get	Modbus Extended Status	UDINT	(See below for definition)
102	0x66	Get/Set	Modbus Type*	USINT	1 = RTU, 2 = ASCII (Default = RTU)
103	0x67	Get/Set	Modbus Slave Address*	USINT	The address of the Modbus Slave (Default = 1)
104	0x68	Get/Set	Modbus Baud Rate*	USINT	The baud rate of the Modbus communication (See below for semantics of values, Default = 7, 19.2 kbps)
105	0x69	Get/Set	Modbus Data Bits*	USINT	7 or 8 Serial Data Bits (Default = 8)
106	0x6A	Get	Modbus Stop Bits	USINT	Always 1
107	0x6B	Get/Set	Modbus Parity*	USINT	(See below for semantics of values, Default = 1, Even Parity)
108	0x6C	Get/Set	Actively Scan Slave*	USINT	0 = No, 1 = Yes (See below for definition, Default = 0)
109	0x6D	Get/Set	Modbus Client Run/Idle Mode*	USINT	0 = Idle, 1 = Run (See below for definition, Default = 0)
110	0x6E	Get/Set	BOOTP Enabled *	USINT	0 = Disabled, 1 = Enabled (Default = Enabled). (See below also)
111	0x6F	Get/Set	EIP Network Status Register*	UINT	A Modbus Register to update with the EIP NS LED State (See below for definition, Default = 0)
112	0x70	Get/Set	EIP Module Status Register*	UINT	A Modbus Register to update with the EIP MS LED State (See below for definition, Default = 0)
113	0x71	Get/Set	Configuration Lock Default on Power-up*	USINT	0 = Unlocked, 1 = Locked. Set to 1 to lock configuration on all subsequent power-ups. Set to 0 to be unlocked on subsequent power-ups. (Default = 0).
114	0x72	Get/Set	Restore Factory Defaults	USINT	0 = No, 1 = Yes. If set to 1, the unit will load factory defaults on next power-up (Default = 0).
115	0x73	Get/Set	IP Address Registers (start address)*	UINT	The first of two consecutive Modbus registers containing IP address. Read on startup if non-zero. (See below for definition, Default = 0)

116	0x74	Get/Set	Subnet Mask Registers (start address)*	UINT	The first of two consecutive Modbus registers containing Subnet Mask. Read on startup if non-zero. (See below for definition, Default = 0)
117	0x75	Get/Set	Gateway Registers (start address)*	UINT	The first of two consecutive Modbus registers containing Gateway. Read on startup if non-zero. (See below for definition, Default = 0)
118	0x76	Get/Set	MAC Address Registers (start address)*	UINT	The first of three consecutive Modbus registers to receive MAC Address. Written on startup if non-zero. (See below for definition, Default = 0)
119	0x77	Get/Set	Modbus Master Inter-scan Delay*	UINT	The amount of time the internal Modbus Master will delay in milliseconds before starting the next loop of reads and writes. (See below for definition, Default = 100 ms)
120	0x78	Get/Set	Modbus Master Inter-message Idle Time*	UINT	The amount of time the internal Modbus Master will delay in milliseconds between the end of receiving a Modbus response and sending the next request. (See below for definition, Default = 10 ms)
121	0x79	Get/Set	Modbus Master Output Heartbeat*	UINT	The amount of time that should elapse before the Modbus Master sends Output data that has not changed. (See below for definition, Default = 0 ms)
122	0x7A	Get/Set	Modbus Master NS/MS LED Heartbeat*	UINT	The amount of time that should elapse before the Modbus Master sends an EIP NS/MS LED status that has not changed. (See below for definition, Default = 0 ms)
123	0x7B		Reserved		

* Denotes that the attribute is saved in non-volatile memory and the value is preserved after a power cycle (unless the “Restore Factory Defaults” attribute was set prior to reset).

Invalid values for a Set_Attribute_Single request return the appropriate EtherNet/IP error response.

3.2.1 Modbus Status (Attribute 100)

This attribute always maintains the current status of the module according to the bit field of flags defined in the table below. A flag is considered to be "Set" if the corresponding bit is 1. If the value of the status attribute is zero (0), then everything is currently operating normally with a valid configuration and no pending edits.

Bit (0 = LSB)	Status Flag
0	Modbus error – slave is responding with error codes. Extended Module Status (Attribute 101) contains the error code information
1	Modbus error – communication with slave timed-out (was previously established)
2	Modbus error – communication with slave cannot be established at all
3-7	Reserved
8	Unit is currently active at factory defaults
9	Unit has a new configuration that will take effect upon reset.
10-15	Reserved

3.2.2 Extended Modbus Status (Attribute 101)

This attribute contains additional error information when a Modbus error in Attribute 100 is flagged. Each byte contains a different value as defined in the table below. Overall, the bytes indicate the error codes that are being returned by the Modbus slave during I/O communication. If multiple Assembly mappings are returning errors, the last mapping's error is contained in this attribute. If the slave stops sending errors, this status will be set back to zeroes.

Byte 3 (MSB)	Byte 2	Byte 1	Byte 0 (LSB)
Modbus Exception Code from slave	Modbus Error Code from slave (0x80 + FC)	Mapping ID receiving the error (1…10)	Assem Inst receiving the error (0x65, 0x66)

3.2.3 Modbus Baud Rate (Attribute 104)

Attr Value	Baud (bps)	Attr Value	Baud (bps)
0	300	8	28800
1	600	9	38400
2	1200	10	57600
3	2400	11	76800
4	4800	12	93750
5	9600	13	115200
6	14400	14	187500
7	19200	15	230400

3.2.4 Modbus Parity (Attribute 107)

Attr Value	Parity
0	None
1	Even
2	Odd

3.2.5 Actively Scan Slave and Run/Idle for Modbus Client (Attributes 108 & 109)

These attributes control the behavior of the Modbus client within the XPort while an EtherNet/IP I/O connection is not present. The Modbus client acts as a “scanner” of the Modbus slave according to the I/O mappings defined in the XPort’s configuration. The client can regularly read the slave’s registers, regularly read and write the slave’s registers, or not communicate at all. The following two tables describe the client’s behavior based on the values of these two attributes:

Value	Attribute 108 – Actively Scan Slave
0	The client will not scan the slave at all when an EtherNet/IP I/O connection is not present. The value of Attribute 109 is ignored.
1	The client will scan the slave when an EtherNet/IP I/O connection is not present. Attribute 109 is used to define the scanning behavior.

Value	Attribute 109 – Run/Idle
0	Idle Mode – the client will only read the slave’s inputs regularly.
1	Run Mode – the client will read the slave’s inputs and write the slave’s outputs regularly.

This functionality can be useful while configuring the device or reading and writing data explicitly via the Assembly Object’s data attribute.

Again, note that these attributes only control the behavior when an EtherNet/IP I/O connection is not established and the configuration is locked. When a connection is established, the client will begin actively scanning the slave, regardless of the value in Attribute 108. The Run/Idle bit in the first word of the output assembly data will tell the client to operate in Idle Mode or Run Mode, regardless of the value in Attribute 109.

NOTE: These attributes must be set to “1” in order to pass EtherNet/IP conformance!

3.2.6 BOOTP Behavior (Attribute 110)

If BOOTP is enabled and the BOOTP request fails (time-out occurs after approximately 30 seconds), then the module will come online at the IP address that resides in non-volatile memory. This will be the value assigned by the previous BOOTP request or the factory default of 192.168.0.254. If BOOTP gets disabled and the power is cycled, the IP address that resides in non-volatile memory will be used.

3.2.7 EtherNet/IP Network and Module Status Registers (Attributes 111 & 112)

These two attributes can be set to contain register addesses in the Modbus slave that should be updated with the EtherNet/IP Network Status and Module Status. If the attribute is set to 0x0000, the XPort will not update anything with the corresponding status. These status registers can be used by the Modbus slave to update LEDs in its hardware. Note that the XPort does not have its own LEDs for this purpose, nor are there enough I/O pins to represent all of the states of two bi-color LEDs. Therefore these LEDs must be implemented in the slave's hardware and firmware. The slave is also responsible for performing the power-on blink sequence and maintaining proper blink timing as defined in the CIP Specification.

The Write Single Register Function Code (0x06) is used by the Modbus client. The valid range of values is 0x0000 to 0xFFFF.

The following table summarizes the actual status values that will be written to the two registers.

MSb					LSb
15	…	3	2	1	0
Reserved			Flash	Red	Green

Bit 0: if 0, green color is off; if 1, green color is on.

Bit 1: if 0, red color is off; if 1, red color is on.

Bit 2: if 0, color is steady/solid; if 1, color is blinking.

For example a blinking green LED would be represented by 101, a solid red LED would be 010, off would be 000, and 011 is undefined since the LED cannot be both colors at once. The remaining 13 bits are reserved for future use, such as reporting additional state information, and should be ignored (not assumed to be zero).

3.2.8 IP Address, Subnet Mask, and Gateway Registers (Attributes 115, 116, & 117)

NOTE: The use of these registers is dependent on whether or not the XPort is set to use BOOTP or not.

These three attributes define the first of two consecutive Modbus registers containing IP Address (attribute 115), Subnet Mask (attribute 116), and Gateway (attribute 117). If the attribute is non-zero (set to a register number), the Modbus master will read/write IP Address, Subnet Mask, or Gateway from/to the Modbus slave's registers. The default for these attributes is zero.

If the attribute is non-zero and BOOTP is disabled:

- On startup, the XPort will **read** the IP Address, Subnet Mask, and/or Gateway from the Modbus registers.
- If the IP Configuration attribute (EtherNet/IP Object Class 0xF5, Instance 0x01, Attribute 0x05) is written with a new valid configuration, the XPort will **write** the new IP Address Subnet Mask to the Modbus registers.
  - If a success response is received from the Modbus write, then success is sent over EtherNet/IP.
  - If an error response is received from the Modbus write, then an error response is sent over EtherNet/IP.
  - A success over EtherNet/IP will cause the XPort to reset and re-read the IP address and/or subnet mask from either non-volatile memory or the Modbus registers.

If the attribute is non-zero and BOOTP is enabled:

- The XPort will not attempt to read the IP Address, Subnet Mask, and/or Gateway from the Modbus registers.
- The XPort will not allow the IP Configuration attribute (EtherNet/IP Object Class 0xF5, Instance 0x01, Attribute 0x05) to be written.
- Upon receiving an IP Address, Subnet Mask, and/or Gateway via BOOTP, the XPort will write the new values to the Modbus registers.

The following table shows an example how the attribute value translates into Modbus register numbers and how the data in these registers translates into an IP address or subnet mask.

Attribute set to (in hex):	Modbus Registers	Register Values	Translates to Address:
“70 00”	40112 40113	0x0201 0x0403	1.2.3.4
“80 00”	40128 40129	0xFFFF 0x00FF	255.255.255.0
“90 00”	40128 40129	0x0201 0x0103	1.2.3.1

3.2.9 MAC Address Registers (Attribute 118)

This attribute defines the first of three consecutive Modbus registers to receive the MAC address of the XPort. If the attribute is non-zero (set to a register number), the Modbus master will write MAC address to the Modbus slave’s registers on startup. The default for this attribute is zero.

The following table shows an example how this attribute value translates into Modbus register numbers and how the data in these registers translates into a MAC address:

Attribute set to (in hex):	Modbus Registers	Register Values	Translates to Address:
“90 00”	40144 40145 40146	0x2000 0x8F4A 0xF290	00:20:4A:8F:90:F2

3.2.10 Modbus Master Timings & Behavior (Attributes 119, 120, 121, & 122)

The Modbus Master inside the XPort has a loop that continually reads and writes the registers of the Modbus slave. These attributes allow you to tune the behavior of the Master and adjust how fast it operates. These can be useful if you want to speed-up the Modbus Master to boost performance or slow it down if your Modbus Slave’s processor cannot keep up.

On every loop, the Modbus Master performs the following in this order:

- Send Network Status LED Message (if necessary)
- Send Module Status LED Message (if necessary)
- Send Modbus Read Messages
- Send Modbus Write Messages (if necessary)

The attributes to control the behavior are defined as follows:

- Modbus Master Inter-scan Delay – The number of milliseconds the Master will delay between every scan loop before executing again. The default is 100 milliseconds.
- Modbus Master Inter-message Idle Time – The number of milliseconds the Master will delay after the end of the receipt of a Modbus response until sending the next Modbus request. The default is 10 ms.
- Modbus Master Output Heartbeat – The number of milliseconds that must elapse before the Master will send a Modbus Write command for output data if it has not changed. The default is 0 ms, which means output data will be written on every scan, regardless of whether it has changed or not. Setting it to 0xFFFF will disable the heartbeat entirely and output data will be written only after a change.
- Modbus Master NS/MS LED Heartbeat – The number of milliseconds that must elapse before the Master will send the Modbus message to write status of the EIP NS/MS LEDs (if configured) if they have not changed. The default is 0 ms, which means the LED status will be written on every scan, regardless of whether it has changed or not. Setting it to 0xFFFF will disable the heartbeat and the LED status will be written only after a change.

The valid range for all of these attributes is 0 to 65535 (0xFFFF) ms.

Note for the heartbeat timings – if the normal execution time of one loop/scan is, for example, 200ms, then setting the heartbeat attribute value to anything between 0 and 200 ms will still cause the message to be sent on every scan.

3.3 Input (T->O) EtherNet/IP to Modbus Mapping (Class 0x65)

The firmware in the XPort maintains a table of the mapping between the EtherNet/IP Input (T->O) Assembly and the Modbus function calls. The information in this table is saved in non-volatile memory. The data in the table can be configured through EtherNet/IP via Get_Attribute_Single (0x0E) and Set_Attribute_Single (0x10) Services directed at Vendor Specific Object Class 101 (0x65), Instance 1.

Attr ID (dec)	Attr ID (hex)	Access	Name	Data Type	Description
100	0x64	Get	Input Data Size	UINT	Current size of the Assembly data in 16-bit words. Whenever a Mapping is changed, this attribute will be immediately updated with the new size.
101	0x65	Get/Set	Mapping 1	Struct of:	
				USINT	Modbus Function Code (0x01, 0x02, 0x03, 0x04)
				UINT	Starting Register Number (Register Address + 1) (0x0001 - 0xFFFF)
				UINT	Quantity of Coils (1-2000), Inputs (1-2000), Holding Registers (1-125), or Input Registers (1-125)
102	0x66	Get/Set	Mapping 2	Struct	Same as Mapping 1
103	0x67	Get/Set	Mapping 3	Struct	Same as Mapping 1
104	0x68	Get/Set	Mapping 4	Struct	Same as Mapping 1
105	0x69	Get/Set	Mapping 5	Struct	Same as Mapping 1
106	0x6A	Get/Set	Mapping 6	Struct	Same as Mapping 1
107	0x6B	Get/Set	Mapping 7	Struct	Same as Mapping 1
108	0x6C	Get/Set	Mapping 8	Struct	Same as Mapping 1
109	0x6D	Get/Set	Mapping 9	Struct	Same as Mapping 1
110	0x6E	Get/Set	Mapping 10	Struct	Same as Mapping 1
111	0x6F		Reserved		

Once a Set_Attribute_Single message is received, the standard EtherNet/IP error checks will be performed. If all error checks pass, the values of the structure will be checked against the set of valid vales. If this check passes, a Modbus message will be sent to the slave device to test if the supplied function code and data are supported in the slave. If a success response is received from the Modbus slave, an EtherNet/IP success response is returned to indicate Modbus success. If an exception response is received from the Modbus slave, an EtherNet/IP error code of 0x1F (Vendor Specific Error) response is sent with the data set to the exact Modbus error response received from the slave (1 byte error code + 1 byte exception code). If there is no response from the slave, the EtherNet/IP error code will be 0x02 (Resource Unavailable).

The mappings are used to construct the Input data message that will be sent from the XPort to the EtherNet/IP connection originator via an I/O message. The input data buffer is organized as follows:

16-bit Status	Mapping 1	Mapping 2	• • •	Mapping 10

The first two bytes provide the status of the device. This value is identical to Class 100, Instance 1, Attribute 100. It provides information about the status of the configuration, health of the device, and status of the Modbus link.

Unused mappings will contain all zeros in the corresponding attribute structure and will not be included in the input data buffer. The input data buffer will be of dynamic total length up to 500 bytes (250 words) according to the combined length of all the mappings. If this length is exceeded, a CIP Error Response 0x1B (Routing Failure, Response Packet Too Large) will be generated for the Set_Attribute_Single message that caused the overflow and the target mapping will remain unchanged.

A mapping can be deleted by setting it to all zeroes.

The factory default configuration does not contain any mappings. The 16-bit status is always provided, so the factory default size of this assembly is 2 bytes.

The data length of the current configuration is in Attribute 100 of this class. The value in this attribute is automatically updated as mappings are changed. Note that the new configuration does not take effect until the power is cycled on the unit. Therefore if the configuration is changed, Attribute 100 will contain the value of the *new* data size that will only take effect after a power cycle - not the size that is currently active.

3.4 Output (O->T) EtherNet/IP to Modbus Mapping (Class 0x66)

The firmware in the XPort will maintain a table of the mapping between the EtherNet/IP Output (O->T) Assembly and the Modbus function calls. The information in this table will be saved in non-volatile memory. The data in the table can be configured through EtherNet/IP via Get_Attribute_Single (0x0E) and Set_Attribute_Single (0x10) Services directed at Vendor Specific Object Class 102 (0x66), Instance 1.

Attr ID (dec)	Attr ID (hex)	Access	Name	Data Type	Description
100	0x64	Get	Output Data Size	UINT	Current size of the Assembly data in 16-bit words. Whenever a Mapping is changed, this attribute will be immediately updated with the new size.
101	0x65	Get/Set	Mapping 1	Struct of:	
				USINT	Modbus Function Code (0x0F, 0x10)
				UINT	Starting Register Number (Register Address + 1) (0x0001 - 0xFFFF)
				UINT	Quantity of Coils (1-1968), Registers (1-120)
102	0x66	Get/Set	Mapping 2	Struct	Same as Mapping 1
103	0x67	Get/Set	Mapping 3	Struct	Same as Mapping 1
104	0x68	Get/Set	Mapping 4	Struct	Same as Mapping 1
105	0x69	Get/Set	Mapping 5	Struct	Same as Mapping 1
106	0x6A	Get/Set	Mapping 6	Struct	Same as Mapping 1
107	0x6B	Get/Set	Mapping 7	Struct	Same as Mapping 1
108	0x6C	Get/Set	Mapping 8	Struct	Same as Mapping 1
109	0x6D	Get/Set	Mapping 9	Struct	Same as Mapping 1
110	0x6E	Get/Set	Mapping 10	Struct	Same as Mapping 1
111	0x6F		Reserved		

Once a Set_Attribute_Single message is received, the standard EtherNet/IP error checks will be performed. If all error checks pass, the values of the structure will be checked against the set of valid vales. If this check passes, a Modbus READ message will be sent to the slave device to test if the supplied registers are supported in the slave. Note: In this case, the XPort will not attempt to write the coils/registers specified by that mapping. If a success response is received from the Modbus slave, an EtherNet/IP success response is returned to indicate Modbus success. If an exception response is received from the Modbus slave, an EtherNet/IP error code of 0x1F (Vendor Specific Error) response is sent with the data set to the exact Modbus error response received from the slave (1 byte error code + 1 byte exception code). If there is no response from the slave, the EtherNet/IP error code will be 0x02 (Resource Unavailable).

Actual writes are only performed when the Modbus client is in Run mode, so any possible errors (i.e. trying to write a read-only register) are reported only when valid I/O is exchanged. The errors are accessible via the status word in the Input I/O data and the status attributes of the Bridge Configuration Object.

The mappings are used to parse the output data message that will be sent from the EtherNet/IP connection originator to the XPort via an I/O message. The mappings are then translated to Modbus data to send to the slave. The output data buffer is organized as follows:

16-bits Run/Idle	Mapping 1	Mapping 2	• • •	Mapping 10

Unused mappings will contain all zeros in the corresponding attribute structure and will not be included in the output data buffer. The output data buffer will be of dynamic total length up to 500 bytes (250 words) according to the combined length of all the mappings. If this length is exceeded, a CIP Error Response 0x1A (Routing Failure, Request Packet Too Large) will be generated for the Set_Attribute_Single message that caused the overflow and the target mapping will be unchanged.

A mapping can be deleted by setting it to all zeroes.

The factory default configuration does not contain any mappings, so the default size of this assembly is 2 bytes.

Only bit 0 of the first word is defined as the Run/Idle command for the Modbus client. When an I/O connection is active, a zero (0) in this bit represents Idle Mode and a one (1) represents Run Mode. In Idle mode, the Modbus client will only read the slave's registers. In Run mode, it will read and write the slave's registers.

The data length of the current configuration is in Attribute 100 of this class. The value in this attribute is automatically updated as mappings are changed. Note that the new configuration does not take effect until the power is cycled on the unit. Therefore if the configuration is changed, Attribute 100 will contain the value of the *new* data size that will only take effect after a power cycle - not the size that is currently active.

3.5 Identity Configuration Object (Class 0x6E)

In order to customize the XPort for different Modbus slaves, the EtherNet/IP Identity Object information must be configurable. This object provides the means to set some Identity Object attributes and the XPort's Ethernet MAC Address. The attributes can be accessed via Get_Attribute_Single (0x0E) and Set_Attribute_Single (0x10) Services directed at Vendor Specific Object Class 110 (0x6E), Instance 1. The attribute mapping parallels that of the standard CIP Identity Object Class 0x01.

Attr ID (dec)	Attr ID (hex)	Access	Name	Data Type	Description
100	0x64	Get/Set	Custom Ethernet MAC Address	6 USINTs	(See below for definition)
101	0x65	Get/Set	Vendor ID	UINT	CIP Vendor ID (Default = 940)
102	0x66	Get/Set	Device Type	UINT	CIP Device Type (Default = 12)
103	0x67	Get/Set	Product Code	UINT	CIP Product Code (Default = 201)
104	0x68	Get/Set	Product Revision	2 USINTs	(See below for definition)
105	0x69		Reserved		
106	0x6A	Get/Set	Serial Number	UDINT	(See below for definition)
107	0x6B	Get/Set	Product Name	Short String	(See below for definition)
108	0x6C		Reserved		

* Note that this entire object will not exist if the firmware is locked.

Invalid values for a Set_Attribute_Single request will return the appropriate EtherNet/IP error response. Any values successfully set will be written to non-volatile memory and take effect after a power cycle.

3.5.1 Custom Ethernet MAC Address (Attribute 100)

This attribute is of the form AA-BB-CC-DD-EE-FF where AA is the first byte in the data buffer and FF is the sixth byte in the data buffer. If this is left as all zeros, the default MAC Address (from Lantronix) of the XPort will be used. The default of this attribute is all zeros.

3.5.2 Product Revision (Attribute 104)

This attribute is a Struct of two USINTs. The first byte in the data buffer is the product's major revision and the second byte is the minor revision. The default is the firmware revision of the XPort.

3.5.3 Serial Number (Attribute 106)

If this attribute is zero, a serial number is constructed from the MAC Address.

3.5.4 Product Name (Attribute 107)

The data type of the Product Name is a "Short String" as defined in the CIP specification. The first byte in the data buffer identifies the length of the product name (number of actual characters, do not count NULL terminators). The second byte is the first letter in the product name; the third byte is the second letter, and so on. Do not include any NULL characters (i.e. string terminator) at the end of the string – only significant characters. The length of the product name is limited to 32 characters. The default product name is "XPort-EIP-MB".

3.6 Input (T->O) Assembly Object (Class 0x04, Instance 0x65)

The mappings defined by Vendor Specific Object Class 0x65 define the standard EtherNet/IP Assembly Object Instance 0x65.

3.7 Output (O->T) Assembly Object (Class 0x04, Instance 0x66)

The mappings defined by Vendor Specific Object Class 0x66 define the standard EtherNet/IP Assembly Object Instance 0x66.

3.8 Configuration Assembly Object (Class 0x04, Instance 0x80)

The Configuration Assembly Object is not implemented. However, some EtherNet/IP clients require one. If this is the case, use Instance ID 0x80 with a data length of 0.

4 TFTP Services

The XPort contains an embedded TFTP server that can be used to read or write a configuration file and update the XPort firmware. However, the firmware must first be unlocked before any files can be uploaded. The configuration file can be downloaded at any time. The TFTP server only supports standard TFTP, not Extended TFTP.

Free TFTP client software is available on the WWW at http://www.weird-solutions.com/

We have recently discovered that the above TFTP client does not behave properly when the server (the XPort) requests retransmission of a packet. Rather than retransmitting the same packet, this client transmits the *next* packet's data using the block number of the packet requested for retransmission. Therefore the server accepts it as the retransmitted data since the block number matches. The result is that the firmware in flash is missing blocks of data. We have observed retransmission attempts occurring only when using the XPort on a half-duplex hub with other network traffic present. The connection is extremely reliable over a full-duplex switch.

4.1 Configuration File

The configuration file contains all of the configurable data within the XPort, as defined in this document. Therefore, rather than having to configure every XPort using EtherNet/IP messaging, a configuration file can be created with a binary/hex file editor and loaded into every XPort.

NOTE: It is NOT recommended to configure the XPort by manually editing this file. Configuration should be performed via EtherNet/IP messaging as described earlier in this manual.

4.1.1 Uploading and Downloading

To download the configuration file, use a TFTP client to connect to the XPort and request the file "config.bin" (without the quotes). Any other file name will not be accepted.

To upload the configuration file, unlock the firmware and upload the file using a TFTP client. The only requirement for the file name is that the file extension is ".bin" – any file name is accepted. Note that on the server, the file name gets changed to "config.bin". The uploaded file is checked for validity before being saved to flash. If the file is valid, the old configuration in flash is overwritten.

4.1.2 Binary File Specification

The following binary file specification for the configuration file is aligned with Firmware version 1.13.

Offset (hex)	Offset (dec)	Type	Num Bytes	Name
0x00	0	UINT	2	Vendor ID
0x02	2	UINT	2	Device Type
0x04	4	UINT	2	Product Code
0x06	6	USINT	1	Major Revision
0x07	7	USINT	1	Minor Revision
0x08	8	USINT	1	Product Name Length, 1-32
0x09	9	32 USINTs	32	Product Name String, 1-32 bytes
0x29	41	UDINT	4	Serial Number
0x2D	45	6 USINTs	6	MAC Address
0x33	51	4 USINTs	4	IP Address
0x37	55	4 USINTs	4	Subnet Mask
0x3B	59	4 USINTs	4	Gateway

Offset (hex)	Offset (dec)	Type	Num Bytes	Name
0x3F	63	USINT	1	BOOTP Enabled Flag
0x40	64	USINT	1	Modbus Type (RTU or ASCII)
0x41	65	USINT	1	Modbus Slave Address
0x42	66	USINT	1	Modbus Baud Rate
0x43	67	USINT	1	Modbus Data Bits
0x44	68	USINT	1	Modbus Stop Bits
0x45	69	USINT	1	Modbus Parity
0x46	70	USINT	1	Active Scan Enabled Flag
0x47	71	USINT	1	Run Mode Enabled Flag
0x48	72	UINT	2	NET LED Register
0x4A	74	UINT	2	MOD LED Register
0x4C	76	USINT	1	Configuration Lock Default
0x4D	77	USINT	1	Restore Factory Defaults
0x4E	78	UINT	2	IP Address Registers Start Address
0x50	80	UINT	2	Subnet Mask Registers Start Address
0x52	82	UINT	2	Gateway Registers Start Address
0x54	84	UINT	2	MAC Address Registers Start Address
0x56	86	UINT	2	Modbus Master Inter-scan Delay (ms)
0x58	88	UINT	2	Modbus Master Inter-message Idle Time (ms)
0x5A	90	UINT	2	Modbus Master Output Heartbeat (ms)
0x5C	92	UINT	2	Modbus Master NS/MS LED Heartbeat (ms)
0x5E	94	UINT	2	Total EIP Input Data Length in Words*
0x60	96	UINT	2	Total EIP Output Data Length in Words*
0x62	98	USINT	1	Input Map 1 – Modbus Function Code
0x63	99	UINT	2	Input Map 1 – Start Address
0x65	101	UINT	2	Input Map 1 – Quantity
0x67	103	UINT	2	Input Map 1 – EIP Input Length in Bytes*
0x69	105		7	Input Map 2 – Same structure as Map 1
0x70	112		7	Input Map 3 – Same structure as Map 1
0x77	119		7	Input Map 4 – Same structure as Map 1
0x7E	126		7	Input Map 5 – Same structure as Map 1
0x85	133		7	Input Map 6 – Same structure as Map 1
0x8C	140		7	Input Map 7 – Same structure as Map 1
0x93	147		7	Input Map 8 – Same structure as Map 1
0x9A	154		7	Input Map 9 – Same structure as Map 1
0xA1	161		7	Input Map 10 – Same structure as Map 1
0xA8	168	USINT	1	Output Map 1 – Modbus Function Code
0xA9	169	UINT	2	Output Map 1 – Start Address
0xAB	171	UINT	2	Output Map 1 – Quantity
0xAD	173	UINT	2	Output Map 1 – EIP Output Length in Bytes*
0xAF	175		7	Output Map 2 – Same structure as Map 1
0xB6	182		7	Output Map 3 – Same structure as Map 1
0xBD	189		7	Output Map 4 – Same structure as Map 1
0xC4	196		7	Output Map 5 – Same structure as Map 1
0xCB	203		7	Output Map 6 – Same structure as Map 1
0xD2	210		7	Output Map 7 – Same structure as Map 1
0xD9	217		7	Output Map 8 – Same structure as Map 1
0xE0	224		7	Output Map 9 – Same structure as Map 1
0xE7	231		7	Output Map 10 – Same structure as Map 1
0xEE	238	USINT	1	8-bit Checksum of all the above values
0xEF	239	USINT	1	1’s Complement of the 8-bit Checksum

Notes regarding some of the fields in the configuration file:

- Input and Output data lengths (marked by *) are re-calculated by the firmware on every boot.
- UINT and UDINT data types are stored in Little-Endian format.
- All arrays or strings of USINTs are stored byte-by-byte in the order the bytes would be visually read from left to right. (i.e. IP Address 12.34.56.78 is stored with 0x12 at the lowest offset going up to 0x78 at the highest offset).

4.2 Firmware Upload

To upload new firmware, unlock the firmware and upload the firmware file using a TFTP client. The firmware file must be a valid DSTni SPB (Serial Program Binary) file with the ".spb" extension. Currently any file name is accepted. The file's first TFTP packet contains the SPB header and checksum information which is validated before any data is written to flash. Note that due to RAM limitations in the XPort, the firmware file must be written to flash as the TFTP packets are received – one by one. Therefore it is imperative that you allow a firmware file to completely upload before resetting the target XPort or closing the TFTP client. Once the header is validated and the upload begins, the old firmware is already being overwritten. Canceling the transfer before completion will result in corrupt firmware that will not load after the XPort is reset. If an error occurs during the transfer, DO NOT reset the XPort. Remedy the problem described by the TFTP error message and attempt the upload again.