



Adding industrial Ethernet with minimal software changes

by John Korsakas

Many industrial products were designed without a growth path to Ethernet networking, but most included serial interfaces. The benefits of migrating to industrial Ethernet interfaces in older and newer products are now compelling. This article discusses the basics of Ethernet/IP and how to leverage off-the-shelf components to implement it.

Proliferation of Ethernet networking in automation systems is on the rise, pushing the vendors of industrial control products such as barcode readers, pneumatic valves, I/O, and so forth, to jump on the networking bandwagon.

To stay competitive in today's automation market, vendors should now be adding intelligent networking to their products. However, implementing support for an industrial network protocol such as EtherNet/IP could take three to nine months to adequately learn the details and confidently write the code. An easier way is needed.

Ethernet expertise not required to start

Enabling technologies for industrial networks including software protocol stacks or hardware components are available today. Essentially, all of the details of the network protocol are already embedded in these components, and one can act as a bridge for the data from your existing product onto the industrial network.

One example is the XPort from Grid Connect. It is a self-contained Serial-to-Ethernet bridge with a 186-based processor, RAM, and flash memory packaged into an RJ-45 connector just about the size of your thumb. The standard XPort supports basic serial tunneling, where data received on each port is

simply retransmitted on the opposite port. Other versions of the XPort support an industrial Ethernet protocol (such as EtherNet/IP and Modbus/TCP), allowing existing products supporting the Modbus RTU/ASCII serial protocol to upgrade from an RS-232/485 serial port to a high performance 10/100 Mbps Ethernet port. Overall, with minor hardware changes and very few software changes, the existing product can be upgraded to support an industrial Ethernet protocol.



Figure 1

Industrial network protocol basics

Industrial protocols have a significant number of differences from standard enterprise networking protocols. Behind every protocol are three basic components:

- Communication Model
- Data Structure
- Services

The Communication Model defines the way data is exchanged between two networked nodes. This includes what can be referred to as *explicit messaging* and *implicit messaging*. Explicit messages are typically one-shot request/response messages used for setting configuration parameters, and implicit messages typically transport the process I/O to and from the device.

Implicit messaging often uses one of the following methods:

- Polling – a one-to-one periodic request and response between a client and a server
- Cyclic – clients and slaves send their data at regular intervals
- Change of state – data is sent only when it changes
- Strobe and multicast – a one-to-many periodic data production, either through an acyclic request and response or cyclic production of data

Other popular communication models are Client/Server and Producer/Consumer.

The Data Structure of a protocol provides a scheme for organizing, storing, and retrieving data within the device. One specific example is the object model within CIP (Common Industrial Protocol), which is the application layer of DeviceNet, ControlNet, and EtherNet/IP. Data is organized into an Object/Instance/Attribute hierarchy. At the highest level are *Objects*, which group together similar types of data such as device identity information, connection parameters, and application-specific data. An Object can then contain one or more *Instances*, for example, a device with eight digital outputs could contain eight instances of the Digital Output Point Object. At the next level are the instance's *Attributes*, which are the specific data values of configuration parameters, application data, and so on. An example of a different data structure is contained within the Modbus Application Protocol, which is used by Modbus RTU/ASCII over a serial line and Modbus/TCP over Ethernet. Modbus data is organized into flat arrays of 16-bit Registers, 1-bit Inputs, and 1-bit Coils (outputs). Each Register, Input, and Coil is assigned an address used to locate the data to be read and/or written.

The organization of data within CIP and Modbus is quite different, but each has its own advantages and disadvantages.

The final component, Services, encompasses the commands and functions that the devices support. Some common services are Read Data, Write Data, Reset, Open Connection, and Close Connection. These may sound simple enough conceptually, but each protocol implements many of them differently, largely due to its underlying Data Structure and Communication Models. For example, to read a data item with CIP, a `Get_Attribute_Single` message must specify the Object ID, Instance ID, and Attribute ID to locate the desired data. The `Get_Attribute_Single` service can read any single attribute within any object, and some devices may additionally implement the `Get_Attribute_All` service to read all the attributes of an object. CIP does not define a service to read only a subset of attributes. Within Modbus, one type of service to read data from a register specifies a base register address where to begin reading and the number of registers to read. Different, but analogous, services are defined for Inputs and Coils. Even the simplest functionality can vary greatly from protocol to protocol.

Mapping serial data to Ethernet

The conversion of serial data from an existing device into Ethernet packets can be trivial depending on the compatibility and similarity of the protocols required on each port. To transparently tunnel the data between the serial port and the Ethernet port, a standard XPort or similar product could easily be used. However, this assumes that the other Ethernet device(s) can parse and understand the format of the data, which may not be the case. If the device's serial port supports the Modbus protocol and connection to a Modbus/TCP Ethernet network is required, luckily the Modbus application layer is very portable from its serial format to its Ethernet format. A clear one-to-one mapping of all the data can be quickly handled by an embedded technology such as an XPort with Modbus Serial and TCP.

EtherNet/IP is one of the more complex industrial Ethernet protocols, and unfortunately it does not have a similar serial counterpart. In this example, consider Modbus-Serial data. As

previously shown, the data structure of EtherNet/IP is object-based. The CIP specification defines numerous standard objects that help make CIP devices more compatible and interchangeable as well as ease the task of programming controllers when installing a CIP network. A very common type of Modbus device is an analog or digital I/O module. Since CIP defines standard Analog and Digital I/O Objects, a mapping should definitely link the Modbus registers, inputs, and coils to these standard objects. For other less common Modbus device types, Modbus data would have to be grouped and mapped into a generic application object.

Packets of EtherNet/IP Process (I/O) data are compiled through CIP assemblies, which are simply data arrays, formed by "assembling" the data from the CIP objects. Modbus inputs would get mapped into CIP input assemblies and Modbus outputs to CIP output assemblies.

The most common Communication Model used in EtherNet/IP is a Producer/Consumer model. Devices "produce" their input data by sending input assemblies to one or many destination nodes at regular cyclic packet intervals. Output data received by devices at cyclic intervals is "consumed" into the devices' output assemblies. The Communication Model of Modbus is a Polled or Request/Response method of exchanging data. A Modbus client regularly sends requests containing output data to a Modbus server, and the server responds with its input data. Note that these two communication models are not exactly compatible: A Modbus server only sends input data when requested while an EtherNet/IP server produces input data at regular packet intervals. Therefore any bridge that converts the data between Modbus and EtherNet/IP must account for this difference.

The XPort's EtherNet/IP-Modbus software allows the product engineer to map the Modbus registers of their existing product into EtherNet/IP objects and I/O assemblies. The configuration of the XPort is done through the Ethernet port via *telnet* and/or web pages and is stored in nonvolatile flash memory. The serial port of the XPort acts as a Modbus client that continually "scans" the Modbus server in the host device. This Modbus client can also be tuned for better performance by

specifying how often to read or write a particular map of the Modbus data as well as assigning relative priorities to each mapping. The mapped data is buffered in a data table within the XPort, which is also read and written by the EtherNet/IP server that handles the communication with the EtherNet/IP client. Once the configuration is complete and tested on one of the XPorts, a configuration file can be downloaded from it and later uploaded to the rest of the XPort-based versions of products during manufacturing. The configuration can also be protected to avoid a problem with customers changing the mappings to something that is not compatible with your device.

Conclusion

Adding an industrial network to a product can be a significant undertaking, but can be easier if enabling components are utilized. This allows an engineering team to quickly implement the networking interface and spend more time focusing on the application of the network in the product's environment. Leveraging the industrial network development tools available on the market and investing time up front to build in-house knowledge and expertise can save your company a great deal of time and money in the long run.



John Korsakas has a BSE and MSE in electrical engineering from the University of Michigan where he was the lab manager of the DeviceNet,

ControlNet, EtherNet/IP, and Modbus Conformance Test Laboratories for five years. During that time he also contributed toward several network, communication, and testing standards for ODVA, ISO, and SEMI. For the last two years John has worked at Grid Connect developing industrial networking software for embedded systems.

To learn more, contact John at:

Grid Connect, Inc.

1841 Centre Point Circle, Suite 143
Naperville, IL 60563
Tel: 630-245-1445 x232
E-mail: johnk@gridconnect.com
Website: www.gridconnect.com