REAL TIME AUTOMATION

PROFINET CBA

An Innovative Distributed Automation Solution

PREPARED BY REAL TIME AUTOMATION

WWW.RTAAUTOMATION.COM

PROFINET CBA

An Innovative Distributed Automation Solution

Introduction

This paper presents an overview of PROFINET CBA, a high-level network for industrial automation applications. Built on standard DCOM and RPC technologies, PROFINET uses traditional Ethernet hardware and software to define a network that structures the task of configuring, accessing and controlling industrial automation devices. PROFINET is based on the object interface structure defined by the Distributed Component Object Model (DCOM) that, based the history of Microsoft inter-process communications, is a descendant of DDE, OLE and Active X.

PROFINET CBA views a system as a series of “Technological Components”. These components act independently and coordinate their activities to form an integrated system. DCOM is a good choice for such a system. It is an object oriented mechanism which structures how a Client (data requester) can locate, request, and transfer data from a Server (data source). Building on the DCOM model, PROFINET CBA strives to provide a seamless integrated system from the sensor-actuator network to the controller and enterprise networks.

PROFINET CBA is much more than PROFIBUS on Ethernet. In fact, the name is very confusing. PROFINET has little in common with PROFIBUS. PROFIBUS is a data oriented mechanism for transferring I/O from device to device over a cable at a maximum speed of 12Meg baud. PROFIBUS data exchange is mainly cyclic with devices that have a prior understanding of the data structure and meaning. In PROFINET CBA, data can be located, information about its structure can be obtained, and data can be exchanged on event triggers, cyclic schedules, or other mechanisms. PROFINET operates on Ethernet at ten to 100 mega baud speeds.

PROFINET uses more traditional Information Technology (IT) concepts and software than all of the other common networks for Industrial Ethernet applications. This has both advantages and disadvantages that will be apparent as the concepts in this paper are developed.

Rocco knows:

PROFINET and PROFIBUS are not the same.

A LITTLE BACKGROUND

Most people who work in an office associate the term “Ethernet” with the physical cable behind their desk. This cable connects their office PC to the printers and Servers of the local network and the infinite web sites on the Internet. This cable is only the physical part of Ethernet, the media carrying Ethernet messages to your PC. On this wire are a whole series of communication protocols such as IP, the Internet Protocol; TCP, the Transport Control Protocol; and various Microsoft protocols such as NetBEUI. This suite of protocols works well for the office environment. It allows users to share files, access printers, send email, search the Internet, and perform all the other communications used in the office environment.

The needs of the factory floor are much different, with some very special requirements. Instead of accessing files and printers, factory floor controllers must access data embedded in drive systems, operator workstations, and I/O devices. Instead of letting a user wait while a task is being performed, factory floor data communications needs are real-time or very close to real time. Terminating the fill operation on a bottle requires much more time-precise communication than accessing the next page of an Internet site.

Traditionally, Ethernet had only limited acceptance in Industrial Automation. Until recently, the expense, lack of sophisticated switches and routers, and domination of large vendors with proprietary protocols prevented the wide acceptance of Ethernet on the factory floor. Now, with prices falling, PCs with inherent Ethernet capability moving in droves onto the factory floor, and intelligent switches and routers, Ethernet is gaining acceptance. Only the lack of a widely accepted, flexible application layer targeted to Industrial Automation has prevented its complete acceptance.

PROFINET COMMUNICATION

PROFINET CBA runtime communication uses TCP/IP, COM, DCOM and RPC (Figure 1). The TCP/IP stack is any standard TCP/IP communication stack. COM and DCOM are Component Object Model technologies as described by the Microsoft standards. RPC (Remote Procedure Call) is a standard communication structure defined by the Open Software Foundation (OSF).

PROFINET CBA does not typically include node configuration and initialization over the network. In fact, vendors are encouraged to maintain all the current proprietary interfaces for configuration to keep the user investment in training and tools. What PROFINET does require is that these tools be revised to include an interface to generate an XML description file. This description file is to include the interfaces present in the device after configuration.

PROFINET also does not preclude integration with other fieldbuses. Specifically, PROFINET suggests that AS-I and PROFIBUS are natural and common extensions to the automation structure through Proxy devices. There is no reason that other sensor and hardware replacement networks cannot be integrated with PROFINET.

DCOM – THE POWER BEHIND PROFINET

PROFINET is based on the Microsoft DCOM (Distributed Component Object Model) architecture and its predecessor COM (Component Object Model). These object distribution architectures encapsulate and manage data and provide external interfaces to the outside world in a fashion that allows Clients and Servers to access data without knowledge of the underlying data structure. COM and DCOM are Microsoft Windows technologies and are found in all Windows versions since Windows 95®.

PROFINET is an object-based technology. Encapsulating automation data into objects is a well-known practice used by other application layer protocols like EtherNet/IP™ and DeviceNet™. Objects allow developers to separate the data implementation from the presentation of the data to the outside world. Object implementation focuses attention on the functionality of the object instead of the minutiae of the implementation. If you are not a software developer, you may not realize how important this concept is.

If you can remember mainframe computers, you remember that data was usually presented as rows of tables. Client programs accessed these mainframe tables from terminal Servers. The terminal Servers presented the data to the user in almost the same format as it was stored. If an additional field was added or data types expanded to offer more resolution or accuracy, a very common occurrence in the early days of mainframe computers, huge numbers of Client programs affecting hundreds of users had to be changed. Separating the data from the interface to the data provided a solution for this type of problem and enabled many other advantages.

Once the interface to the data was separated from the data, the interfaces could be linked between Clients and Servers using any number of technologies. Some of the mechanisms used prior to COM included:

- **TCP – A connection-based transport protocol. TCP does not directly support object-based data. Instead it supports simple byte-streams.**
- **UDP – A connectionless-based transport protocol with the same limitations as TCP.**
- **Windows Messaging – Microsoft Windows-based mechanism for communicating data between processes in a single machine. Messaging is, by definition, limited to a single computer.**
- **DDE – Dynamic Data Exchange – A connection-based protocol with a distinct set of messages for sharing data between processes in a single machine. DDE is part of the windows messaging system.**
- **RPC (Remote Procedure Call) – A mechanism created by the Open Software Foundation (OSF) for sharing data between Client and Server processes using standard interfaces. RPC, unlike some other technologies, is not limited to windows applications. RPC is also not limited to a particular transport mechanism. RPC can ride on TCP, Microsoft Pipes, and other networking systems. The limitation of RPC is that it links processes at a function-call level, not at an object level.**

COM and DCOM are successors to all these technologies. COM is a language independent technology that offers object linking, technology not found in RPC or any of the other mechanisms for sharing data. Where these mechanisms share code and data, COM exists to share objects between processes. Where COM shares objects between processes local to an operating system, DCOM shares objects between processes located across a network.

It is important to understand that COM is not a language. COM is a specification which describes what an object is, when and how it gets created, and how it tells the world what it can do. COM groups functions together into an interface, which can be named and registered. No particular implementation of the interface functions is implied.

COM interfaces can be thought of as a contract between a Client and a Server. COM provides interfaces which allow a Client to query a Server and discover the interfaces that it supports. There are a number of standard interfaces that all COM Servers must support. The most well known of these is the IUnknown interface. This interface provides the query function that returns the IDs of data interfaces supported by the object.

DCOM extends COM by including all the functionality to distribute data across a network in a reliable, secure, and efficient manner using the same objects used by COM. DCOM provides the network interfaces, automatically packages and unpacks data in the native machine format, and simplifies the management of initiation and termination of Server processes. It also provides a robust security mechanism with inherent encryption of data.

THE KEY TO UNDERSTANDING PROFINET - PROFINET INTERFACES

PROFINET interfaces, how PROFINET communicates with the outside world, are Component Object Model (COM) interfaces. There are two parts to a COM interface; the interface and the implementation. The interface is the set of access methods supported by the object, while the implementation is the underlying structure of the object. The interface is public to the world, while the implementation is private to the developer. The implementation can change at any time while, the interface must remain consistent to provide support for existing applications.

A COM interface is a collection of Properties, Methods, and Events. Properties are the public data of a PROFINET CBA object. These values can be read or written from a remote device. Methods define the services and functions provided by the COM interface. For example, a barcode scanner can have a method that starts a scan; a drive can have a high-level method to position a roller; or an Analog Input module could have a method to set the resolution of its input. In addition, COM interfaces have Events. Events signal a change in state for the interface and can be used to invoke a method in a Client. The Interface Definition Language (IDL) is a structured, textual representation of the Properties, Methods, and Events supported by a PROFINET object.

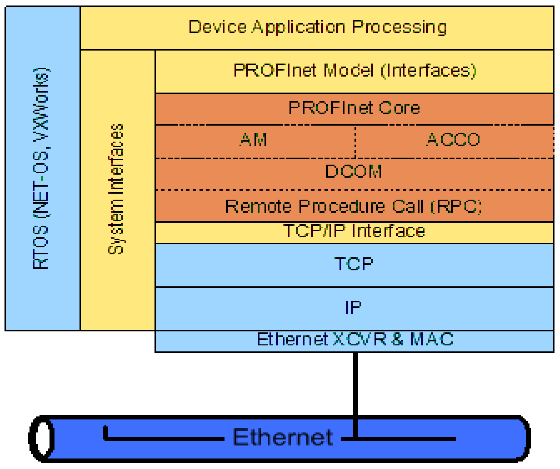
Every COM interface contains the IUnknown interface. This interface is the root interface of all COM objects. All other COM interfaces are derived from IUnknown or from another interface that is derived from IUnknown. A Client device can discover the entire interface set for a PROFINET device by requesting the interfaces supported by IUnknown. IUknown provides QueryInterface as the method to discover interfaces.

For more information on PROFINET interfaces, read any of the large volume of text books describing COM and DCOM.

PROFINET RUNTIME STRUCTURE

The PROFINET Runtime Software or “PROFINET Core” is the software made available by PROFIBUS International to developers. This software is only one of a number of components necessary to implement a PROFINET device. See Figure 1 for a graphical representation of the components of a PROFINET device.

Figure 1 — PROFINET Runtime Software Components



Blue indicates off-the-shelf components supplied by the developer. Yellow indicates integration tasks for the developer. Orange indicates parts of the PROFINET core that a developer should not change.

TCP/IP Stack

The Transport Control Protocol (TCP) and Internet Protocol (IP) are the software components which make connections between devices, assemble and disassemble packets, manage connections, and route packets from one endpoint to another. The TCP/IP stack is not included in the PROFINET Runtime Core. It is usually an off-the-shelf software component and is often included with the RTOS (Real Time Operating System). PROFINET includes, by default, function calls to the Berkley Sockets Interface and NT Winsockets. Other TCP/IP stack interfaces require more extensive integration efforts.

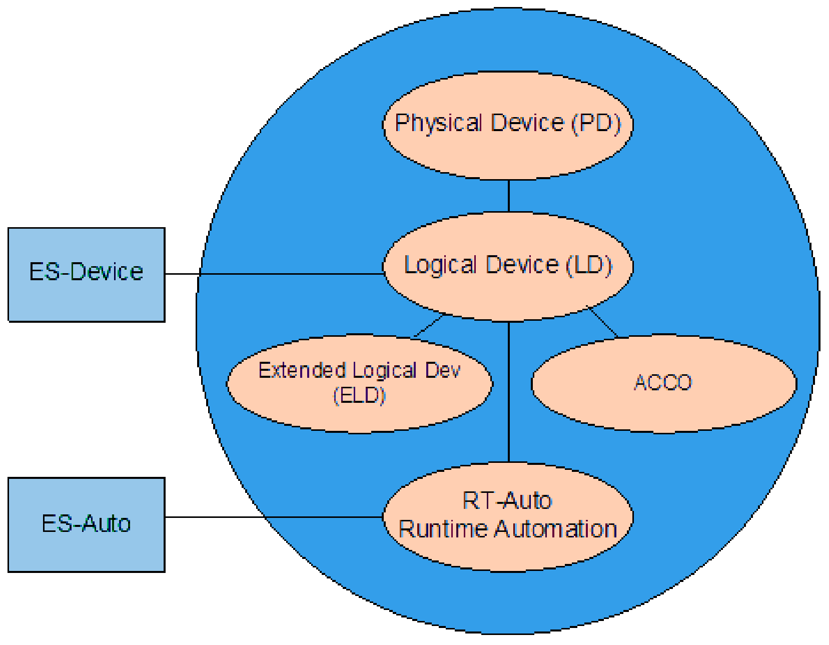
Remote Procedure Call (RPC)

The Remote Procedure Call software transports DCOM requests between a PROFINET core and the TCP/IP stack. The RPC is a standard component of Microsoft Windows. For embedded systems running a non-windows operating system, the PROFINET core includes a standalone RPC. The RPC provides a consistent method of making remote procedures calls in a PROFINET system, and it uses the connection-based messaging of the TCP/IP stack and memory management, task synchronization and critical resource management services of the RTOS. To integrate RPC into a PROFINET device, both the interface to TCP/IP and the RTOS must be customized. If procedure tracing is implemented by the PROFINET device, the RPC must be customized to include the standard interfaces to the diagnostic services.

Distributed Component Object Model (DCOM)

The Distributed Component Object Model software manages the distribution of the properties, methods, and events of the interfaces included in a PROFINET device. The DCOM is a standard component of Microsoft Windows. For embedded systems running a non-windows OS, the PROFINET core includes a standalone DCOM. The DCOM component must be integrated with four other components; the RPC, the RTOS, the DCOM Application Interface and the Diagnostic module. The standalone DCOM provided by PROFIBUS International contains the standard RPC API, so little effort is required to integrate DCOM and RPC. The integration of DCOM and the services provided by the RTOS is much more complex. Since DCOM calls operate asynchronously, the RTOS services use callback routines, necessitating a highly customized interface between the RTOS and DCOM.

Figure 2 — PROFINET Object Model



Auto Marshaler (AM)

The Auto Marshaler serializes the components of a parameter call prior to handing the marshaled call to DCOM. Marshalling on the Client side includes placing parameters in a message and converting the local data type to the data type of the network RPC. Marshalling on the Server side includes removing parameters from the message and converting the data type into the data type of the local Server. The Auto Marshaler performs both Client and Server operations. When the Auto Marshaler initiates a method in an interface, it functions as a Client. When the Auto Marshaler receives access to a method, it functions as a Server. Auto Marshaling insulates the local Client or Server from the details of the call interface. Clients and Servers can call all methods as if they are local methods. Auto Marshaling is part of DCOM and is highly integrated with the DCOM provided in the PROFINET core. Integration activities for this component include RTOS and Diagnostic services integration.

PROFINET Object Model

A PROFINET Application is a device-specific implementation of the PROFINET Runtime Object Model. The objects that exist in a device, their methods and interfaces accessed externally through COM Automation, comprise the Runtime Object Model (Figure 2). The basic components of a Runtime Object Model are:

- **Physical Device (PD) - A representation of the physical hardware that connects the device to the Ethernet network. The PD is exposed by the IPhysicalDevice Interface.**
- **Logical Device (LD) – A representation of the application that implements the task of the device. The LD is specific to the task of this device.**
- **Extended Logical Device (ELD) – A representation of the generic tasks in common between devices of the same class.**
- **Active Control Connection Object (ACCO) – A representation of the PROFINET software that facilitates automation through RT-Auto.**
- **Runtime-Auto (RT-Auto) – The entire collection of methods and interfaces exposed to the outside world by the PROFINET device**
- **Engineering Service Auto (ES-Auto) – The collection of methods and interfaces needed for the Engineering Model to access the methods and interfaces provided by RT-Auto.**
- **Engineering Service Device (ES-Device) – The collection of methods and interfaces needed for the Engineering Model to access the methods and interfaces provided by ELD. These methods and interfaces are common between like devices.**

The PROFINET Object Model provides a standard interface to all Clients to monitor and control the physical characteristics of a connection. There is one physical connection for each hardware component. The PD provides the resource to all logical connections. Using the known PD interface all other interfaces for all logical units can be discovered.

There are one or more of these logical connections for each physical connection on the PROFINET network. A logical connection represents a unit of firmware and the sensor or actuator part of the device.

Active Control Connection Object (ACCO)

The Active Control Connection Object is a software component in the PROFINET Runtime Software core. The ACCO manages cyclic and Change-of-State (COS) data transfer operations. Cyclic data transfer occurs on a timed basis, controlled by the ACCO Quality Of Service (QoS) method. At present, COS data transfer operations are planned for a future release of PROFINET. One device's ACCOs communicate with ACCOs of other devices to manage data transfer, and ACCOs can operate in a local device or in a remote device across a network connection.

PROFINET ENGINEERING MODEL

PROFINET terms the process of building a PROFINET device, creating interconnections between the PROFINET devices of a system, and sending the resulting runtime configuration to the system the “PROFINET Engineering Model”.

PROFINET devices are built by decomposing the functionality of the device into components. Each component is modeled as a set of properties, events, and methods. Once these external “interfaces” are designed, an eXtensible Meta Language (XML) file is created. The XML file captures the functionality of the device in a text-based, transportable image that can be processed by the PROFINET Connection Editor.

The PROFINET Connection Editor is a device used to do the System Engineering of a network. The Connection Editor provides a graphical tool for choosing the relationships between devices on a PROFINET network. Connections are made between devices at the component layer. An event in one device can trigger a method in another interface. It makes no difference if the method is part of the same interface or a remote device located miles or countries away.

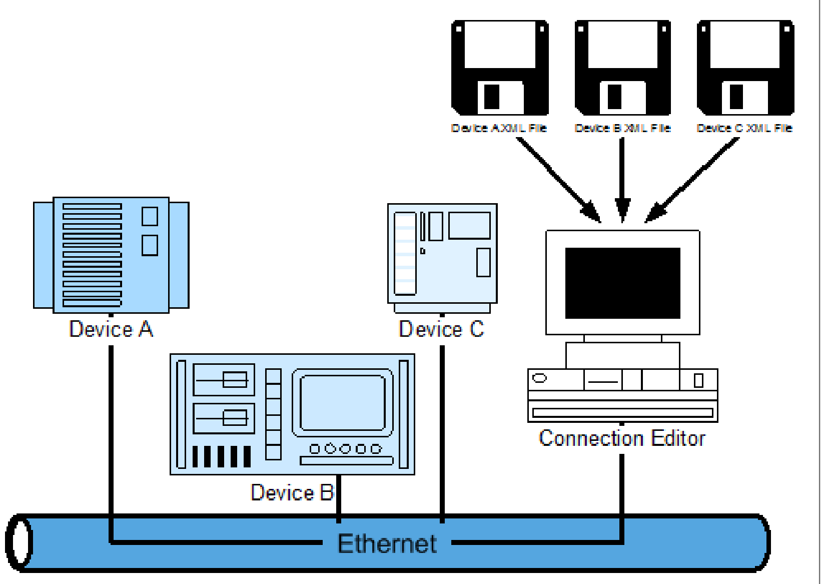
Once the relationships are chosen, the Connection Editor automatically downloads the connections to each PROFINET device. These connections are stored in non-volatile memory and become the basis of the PROFINET Runtime operation. The connections describe to each device what TCP/IP connections must be made, what Remote Procedure Calls are required, and what DCOM interfaces are triggered and when they are triggered.

WHERE DOES PROFIBUS FIT IN?

PROFIBUS® is a worldwide standard for transmission of process data between field level devices and programmable controllers. How can I/O data and other information travel between PROFIBUS and PROFINET? At the physical level, these two communication networks are completely incompatible. PROFIBUS is, at its core, an RS485-based communication standard with a maximum baud rate of 12M. At the core of PROFINET is Ethernet, another worldwide standard with baud rates of 100M and beyond.

Are they at all compatible at the protocol level? PROFIBUS is I/O driven. Data is transferred as blocks of inputs and outputs which are interpreted by agreement between the PROFIBUS Master and the Slave. PROFINET is an object oriented system where any interface can be connected to any other interface. There is no hard and fast definition of a Client or Server. A device interface can be configured to supply data as a Server or consume data as a Client. Connections are made at system integration time using the connection editor at an object level between device interfaces.

Figure 3—PROFINET Engineering Model



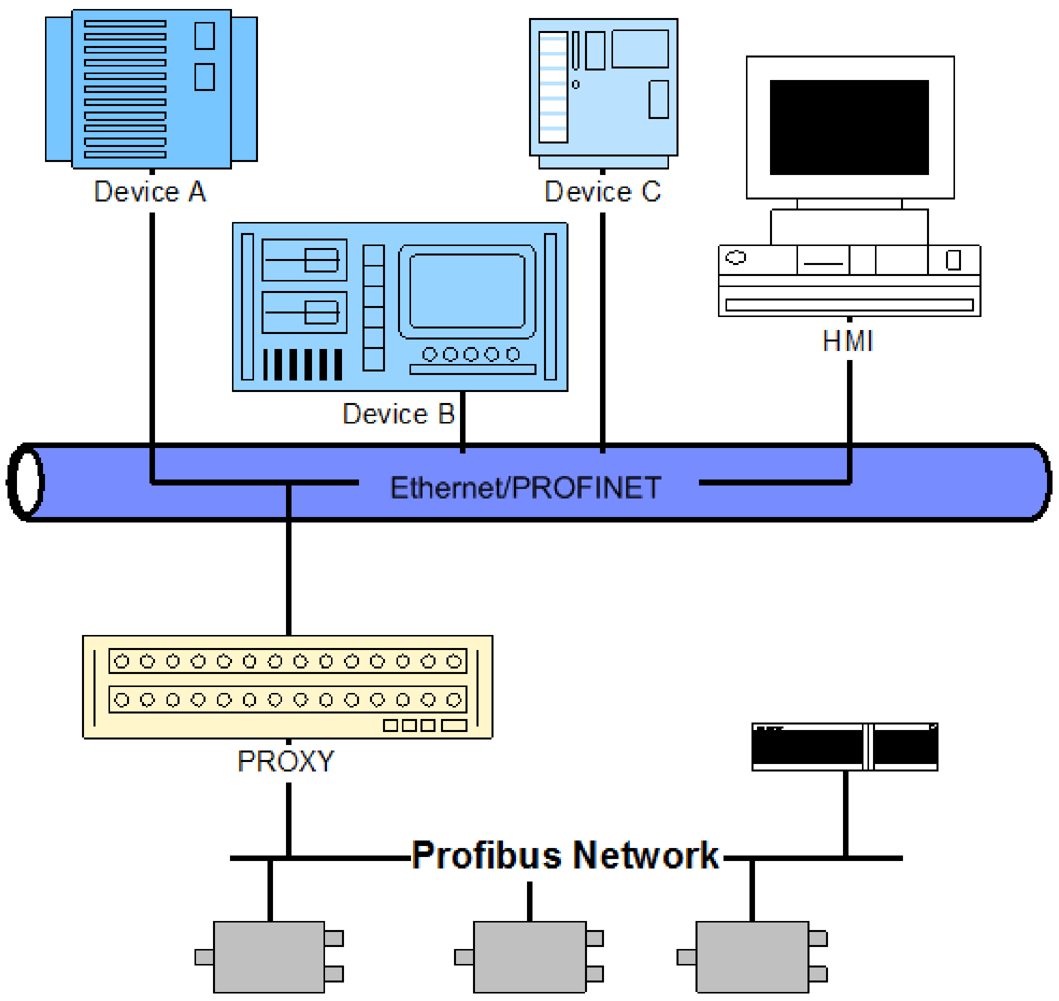
Connections form the sensor bus-level to Enterprise Resource Management (ERP) and Manufacturing Execution System (MES) systems require PROFIBUS to PROFINET Gateways. Since the PROFIBUS Master device can access all the data of the PROFIBUS network, the logical place for this gateway is the Master device, which is usually the programmable controller. These devices will most likely be equipped with both technologies and it is reasonable to expect that this is where the link in these systems will be found. (Figure 4)

PROFINET CERTIFICATION

Unlike PROFIBUS, all PROFINET devices must pass certification prior to product launch. The Certification Office of the PROFIBUS User Organization is responsible for establishing test centers where PROFINET devices can be certified.

Certification of a PROFINET device is a three step process. First, the parts of the PROFINET software core modified by the developer and the PROFINET Object Model are tested for compliance with standards. Second, the XML file supplied by the product developer is used with an Engineering tool to test interoperability. In this test, connections are made between the interfaces of the Device Under Test (DUT) and other PROFINET devices. Third, the PROFINET device is connected to a network of other PROFINET devices to determine how well the device works on a loaded network.

Figure 4—PROFIBUS Proxy Operation



HOW TO GET STARTED

The PROFINET Runtime Core software can be downloaded free of charge (membership required) from PROFIBUS International as a zip file. Porting PROFINET to a specific RTOS requires the developer to port each PROFINET component. The components to port include the DCOM, RPC, Auto Marshaller, and the ACCO. The PROFINET Integration manual provides individual instructions for porting each of these components.

Prior to starting this effort the developer should have a background in the Microsoft RPC, COM, DCOM and C++. An extensive background in the specific operation of your RTOS, the operation of your TCP/IP stack, and the fundamentals of your processor is also important.

PROFINET implementation is not without challenges. Porting of your PROFInet core to your embedded RTOS is a significant challenge. The PROFINET core is designed for windows. It is a major effort to port the PROFINET core to a non-windows RTOS. Some of the more challenging aspects of this integration effort include mapping the Windows memory management services, real time clock functions, and DCOM to the embedded system. The target device requires significant resources. The PROFINET core contains a large number of source files with significant code and memory requirements including non-volatile memory.

SUMMARY AND PERSONAL OBSERVATIONS

There are numerous application layer competitors to PROFINET including MODBUS TCP/IP® from Groupe Schneider, EtherNet/IP™ from the Open DeviceNet Vendor Association (ODVA), HSE Fieldbus from the Fieldbus foundation and other networks from other vendors. PROFINET distinguishes itself from its competitors through the depth of integration in standard Information Technology (IT) functionality. Where its competitors are industrial protocols riding on TCP/IP, PROFINET is all IT with a smattering of industrial automation.

Is that an advantage or a disadvantage? This embracing of a Microsoft standard certainly makes it more attractive and easier for IT professionals to develop industrial applications. The MES and ERP systems will be very easily integrated with PROFINET networks. Traditional automation vendors with PC-based architectures will also enjoy an advantage as they have staff trained in this technology. Other vendors of embedded devices like drives, scales, and barcode scanners will face a steep learning curve to porting and implementing this technology in their products.

The Microsoft factor must also be mentioned. Microsoft obsoletes technologies on a daily basis. How long will DCOM and RPC survive? What happens when future Microsoft Windows products reduce support for these products and increase support for others. In fact, SOAP is already being touted as a replacement for RPC. In an industry requiring technology support over decades is a strategy requiring support from Microsoft viable?

FOR MORE INFORMATION

- **PROFIBUS International, *PROFInet Architecture Description and Specification*, Version 1.0, August 2001**
- **PROFIBUS International, *PROFInet Implementation Guide*, Version 1.2, July 2002**
- **Open Software Foundation (OSF), www.opensoftwarefoundation.com**
- **Dr. Richard Grimes, *"DCOM Programming"*, 1997 Wrox Press Ltd.**

Appendix 1: PROFINET Terms

PROFINET, like many other networking systems, has a set of unique terminology. The following table contains a few of the terms commonly used when discussing PROFINET.

Term	Abbreviation	Definition
Auto Marshaller	*AM*	Software which packages function calls and parameters to be sent over COM
Component Object Model	*COM*	Microsoft derived mechanism to distribute objects between processes within a single machine.
Distributed Component Object Model	*DCOM*	Microsoft derived mechanism to distribute objects across a network.
Component Builder	*Componer*	An add-on to a vendor tool that generates XML files which are used to connect interfaces between objects at the PROFINET system level
Core		PROFINET Core software provided free of charge by PROFIBUS International (PI)
Engineering Model		PROFINET term for process of creating relationships between interfaces of PROFINET devices
Global Unique Identifier	*GUID*	Identifier used by COM to uniquely identify an object and interfaces
Interface Definition Language	*IDL*	Script Language used to describe interfaces between PROFINET devices
Internet Protocol	*IP*	Routing protocol running under TCP
Object Linking Editing	*OLE*	Precursor to COM
PROFIBUS International	*PI*	
Quality of Service	*QoS*	Specifies how often a value is transferred over a network interface
Remote Procedure Call	*RPC*	An API for making function calls between a Client and a Server. Microsoft RPC is compliant with the Distributed Computing Environment (DCE) developed by the Open Software Foundation (OSF). RPC is the protocol running under DCOM.
Real Time Operating System	*RTOS*	The non-windows based control software in a PROFINET device.
Runtime		PROFINET application software that transfers data between physical devices as COM objects
Transport Control Protocol	*TCP*	Transport communication software running under RPC. TCP provides connections
Universal Unique Identifier	*UUID*	Same as GUID – See GUID
eXtensible Markup Language	*XML*	XML is a simple, very flexible text format originally designed to meet the challenges of large-scale electronic publishing.