OPC The " Enabling" Software

Marilyn Self
GE Sales Project Specialist

Today's demands on software interfacing capabilities far exceeds anything imagined when the market began supporting Windows' dynamic data exchange (DDE) programming interface for communication with devices. This paper will provide an insight into these developments and describe the new king on the block, OPC.

What is OPC?

OPC is an industry standard created by numerous automation hardware and software suppliers around the world working with Microsoft. OPC stands for OLE (Object Linked Embedding) for Process Control. To insure "OPENNESS", the OPC Foundation, consisting of over 220 members, manages the OPC standard. The OPC standard is based on Microsoft's OLE (now Active X), COM (component object model) and DCOM (distributed component object model) technologies. Basically, OPC consists of a standard set of interfaces, properties, and methods that uses modules and objects (see list of links at the end of the document for added detail) to solve application problems. The standard consists of clients and servers, where the OPC clients initiate communications to the OPC servers and the servers respond.

The OPC standard defines a technology not a product. The OPC goal is "to provide an open, flexible, plug and play software standard for software interoperability in the automation industry." The stated benefits of OPC are:

- Hardware manufactures will only have to make one set of software components for customers to be able to utilize their applications.
- Software/application developers will not have to rewrite drivers because of feature changes or additions in a new hardware/firmware release
- Customers will have more choices with which to develop a best in class integrated manufacturing system.

An OPC server is software that translates between hardware's language, such as ModbusRTU, and the application that needs the data. It is a communication standard where all devices agree on the format of the data, sent and received.

Servers do not initiate a dialog. They respond to a request from a Client asking for data. Typically OPC Servers communicate to devices and interfaces. Applications use OPC Clients to communicate to OPC servers. The OPC client and server could, and usually does, reside on the same machine. The link between the OPC server and the device is the driver. It can be serial, Ethernet, IR or radio waves, it doesn't matter. The application only finds the OPC server and the server gets the data from the device and responds back to the client.

In the past, client applications have developed customer drivers for their specific application, resulting in the following issues:

- Application software must write a driver for every piece of hardware used
- All hardware features are not supported by all drivers
- A change in the hardware's capabilities requires a new driver
- Device access conflicts

OPC is about standardization, consistency and commonality.

Terminology needed

COMMUNICATION – All devices agree on the format of the data.

COM (COMPONENT OBJECT MODEL) – defines software architecture to build component-based application on. COM objects each have a unique identity. They provide interfaces allowing applications and other components to access their unique features. COM objects are completely language-independent (unlike WIN32 DLLs). They have IPC capability built-in. COM was first released in 1993 to replace the IPC mechanism of DDE used in the initial release of OLE.

DDE (DYNAMIC DATA EXCHANGE) – an IPC (interprocess communication) system built into operating systems that enables two running applications to share the same data

DCOM (Distributed Component Object Model) – based on COM, DCOM supports the same model of component interaction, but supports distributed interprocess communication. Thus DCOM is an architecture that enables components (processes) to communicate across a network.

DRIVER – translator between a device and any programs using the device. The driver (a piece of software) accepts generic commands then translates the accepted commands into specialized commands for the device.

IPC (INTERPROCESS COMMUNICATIONS) – allows one process to communicate with another process.

OLE (OBJECT LINKING AND EMBEDDING) – creates objects with one application then allows the linking or embedding of the object in another application. This support is built into the Windows operating system.

PROTOCOL – is an agreed upon format for transmitting data between two devices. The main items are:

- Type of error checking
- Data compression
- How the sending device will indicate it has finished sending data
- How the receiving device will indicate that the message has been received

Why is the market migrating from DDE to OPC servers?

DDE lacks the detail data structure needed for interfacing information in today's more sophisticated applications since it was never intended for such use. As a result, client applications requiring data from specific devices have had to independently develop custom drivers for each. This has led to problems such as hardware features, which were not supported, by all driver developers, device access limitation, changes in hardware, which could "break" some drivers. OPC inherently differentiates between hardware providers and software applications.

In addition, Microsoft has not changed the DDE protocol since about 1990 and stopped updating the DDE interfaces of applications where COM does the job better. Most important is whether Microsoft will continue to support DDE communication with its applications. Perhaps not, but they will hopefully never remove the existing DDE interfaces for backwards compatibility, although changes and degradation have already happened.

A Common Example of Days Past – the Printer Driver

In the 1980's when MSDOS was the preferred operation system for PC's, each application provided a driver for every printer on the market. This was also the case on the industrial automation front where each HMI (Human Machine Interface), such as Intellution, Cimplicity, and Wonderware provided a proprietary driver to each printer device supported. The natural impact was that only supported printer devices could be used by a given application and any changes to the software or hardware could result in the printer no longer being supported. A variety of printers were often required in order to print from different applications. Even new printers held no guaranties. If they did work, it could be on a very limited basis. The user, the printer manufactures and the application developer shared frustration.

Microsoft solved the printer problem with the introduction of Windows. They provided printer support in the operating system, thus removing the printer support from the applications developer. The drivers were now supplied by the printer maker who had the biggest interest in making them work, not the applications developer. This was a solution that worked for the end user, the printer manufacture and the application developer.

An Industry Example of How Drivers Applied Before And Using OPC; Before:

Our example plant uses an HMI, such as Cimplicity, a Process History Machine and a Condition Monitor, all needing information from different devices such as a PLC, an EPM9650 meter, and a calculation engine. The PLC might be using the Modbus + protocol, the EPM9650 with Modbus over TCP/IP and the calculation engine might be only DDE compliant. Prior to OPC, each application would require a custom driver to talk to

each device and each device would have to respond to each application. This would be called a traditional propriety/open system, illustrated in Figure 1. Such a system would require nine custom drivers, one from each application for each device. Multiple data requests per device would, to each device, would slow the system considerably.

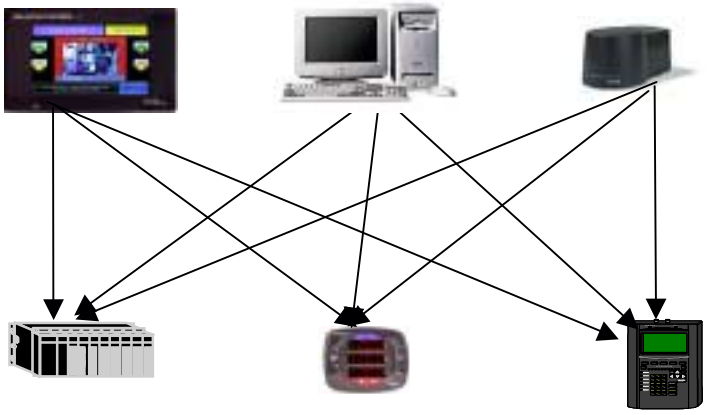

Figure 1

The Industrial Example Using OPC

Using the same scenario as previously, a plant using an HMI, a Process History Machine and a Condition Monitor, all needing information from devices such as a PLC, an EPM9650, and a calculation engine. The PLC using the Modbus + protocol, the EPM9650 using Modbus over TCP/IP and the calculation engine using only DDE, and using an OPC solution. The requirement would be 3 OPC servers and 3 OPC clients, one for each protocol type. Each application through its OPC client would make the call to the OPC server; the OPC server would then make one call to the device. All interfaces would be standard "off the shelf", thus eliminating custom/proprietary drivers. Instead of nine calls to the device each device would only receive three calls (figure 2).

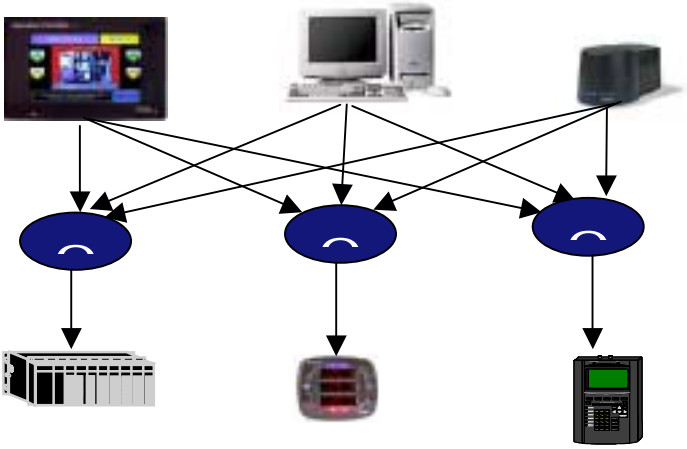

Figure 2

.

Conclusion

OLE for Process Control draws a line between hardware providers and software developers. It provides an effective mechanism to quickly access data from devices and communicate the data to any client application. It provides a standard communications layer between industrial devices and applications. This allows the user to choose best-in-class products, both hardware and software – PLC's, Operator Interfaces, meters, calculation engine and HMI packages – that can communicate with OPC instead of serial/custom drivers. Providing the server with an OPC interface allows any client to access their devices, using a mechanism, which Microsoft will continue to support into the future.

References and Links

The OPC, OLE for Process Control Product Overview, Version 1.0, OPC Foundation 1998.
Roger Sessions, COM+ and the Battle for the Middle Tier, John Wiley & Sons, Inc., 2000.
Rosemary Rock-Evans, DCOM Explained, Butterworth-Heinemman, 1998.
Gilbert Held, Understanding Data Communications, Second Edition John Wiley & Sons, Inc., 1996.
Kevin Dowell, OPC – The Simple Truth, GE Fanuc – Cimplicity Business, White paper, 2001.

http://www.opcsource.com/
http://www.matrikon.ab.ca
http://www.opcfoundation.org/
http://opcactivex.com/