5-1-2002

Effectiveness of OPC for systems integration in the process control information architecture

Philips Zachariah

Effectiveness of OPC for Systems Integration in the Process Control Information Architecture

Philips Zachariah

B.E. (Instrumentation)
(Mumbai University, Mumbai 1999)

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the Department of Industrial and Systems Engineering in the Kate Gleason College of Engineering of the Rochester Institute of Technology

May 2002

KATE GLEASON COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

MASTER OF SCIENCE DEGREE THESIS

The M. S. Degree Thesis of Philips Zachariah
has been examined and approved by the thesis committee
as satisfactory for the thesis requirement for the
Master of Science degree.

Dr. Sudhakar Paidy, Ph.D. *Advisor*

Dr. Mathew Marshall, Ph.D.

Permission granted

Effectiveness of OPC for Systems Integration in the Process Control Information Architecture

I, Philips Zachariah, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: 05/22/02 Signature of Author: _____

Dedication

To my family and my relatives who taught me that
"Fear of the LORD is the beginning of Wisdom"

Acknowledgment

Throughout the time I have been working in this research study, many people have assisted me. It is impossible to acknowledge them all, nevertheless, I would like to specifically thank the following individuals:

Dr. Sudhakar Paidy, for all the time he dedicated to my research work and for the knowledge he has shared with me in the last two years. This thesis would not have been possible without your help.

Dr. Mathew Marshall, for his support and comments on my work. I hope you enjoyed being a member of my thesis committee.

The faculty and staff of the Industrial and Systems Engineering Department, not only for their academic support and feedback on my work, but for all the advice given throughout my stay at RIT.

To Mr. Dennis Wilk and Mr. Barry Hawley of Siemens for their constant help and support making my thesis work a reality.

To my wife Reena Abraham for her support, aiding me in my thesis completion.

To all my friends who helped me from the beginning of the thesis for giving me their feedback and support. But mostly, for your friendship, all of you have a special place in my heart.

Outline

Section **Page**

Tables and Figures

Abstract

A Process is defined as the progression to some particular end or objective through a logical and orderly sequence of events. Various devices (e.g., actuators, limit switches, motors, sensors, etc.) play a significant role in making sure that the process attains its objective (e.g., maintaining the furnace temperature within an acceptable limit). To do these things effectively, manufacturers need to access data from the plant floor or devices and integrate those into their control applications, which may be one of the "off the shelf" tools such as Supervisory Control and Data Acquisition (SCADA), Distributed Control System (DCS), or Programmable Logic Controllers (PLC). A number of vendors have devised their own Data Acquisition Networks or Process Control Architectures (e.g., PROFIBUS, DEVICENET, INTERBUS, ETHERNET I/P, etc.) that claim to be open to or interoperable with a number of third party devices or products that make process data available to the Process or Business Management level. In reality this is far from what it is claimed to be. Due to the problem of interoperability, a manufacturer is forced to be bound, either with the solutions provided by a single vendor or with the writing of a driver for each hardware device that is accessed by a process application. Today's manufacturers are looking for advanced distributed object technologies that allow for seamless exchange of information across plant networks as a means of integrating the "islands of automation" that exist in their manufacturing operations. OLE for Process Control (OPC) works to significantly reduce the time, cost, and effort required in writing custom interfaces for hundreds of different intelligent devices and networks in use today. The objective of this thesis is to explore the OLE for Process Control (OPC) technology in depth by highlighting its need in industry and by using the OPC technology in an application in which data from a process controlled by Siemens Simatic S7 PLC are shared with a client application running in LabVIEW6i.

1. Introduction

Industrial automation involves gathering plant related data from the factory floor (with the help of field devices) to higher levels of the Process Control Information Architecture. The Process Control Information Architecture consists of three layers, namely: [1]

1. Field Management Layer,
2. Process Management Layer, and
3. Business Management Layer

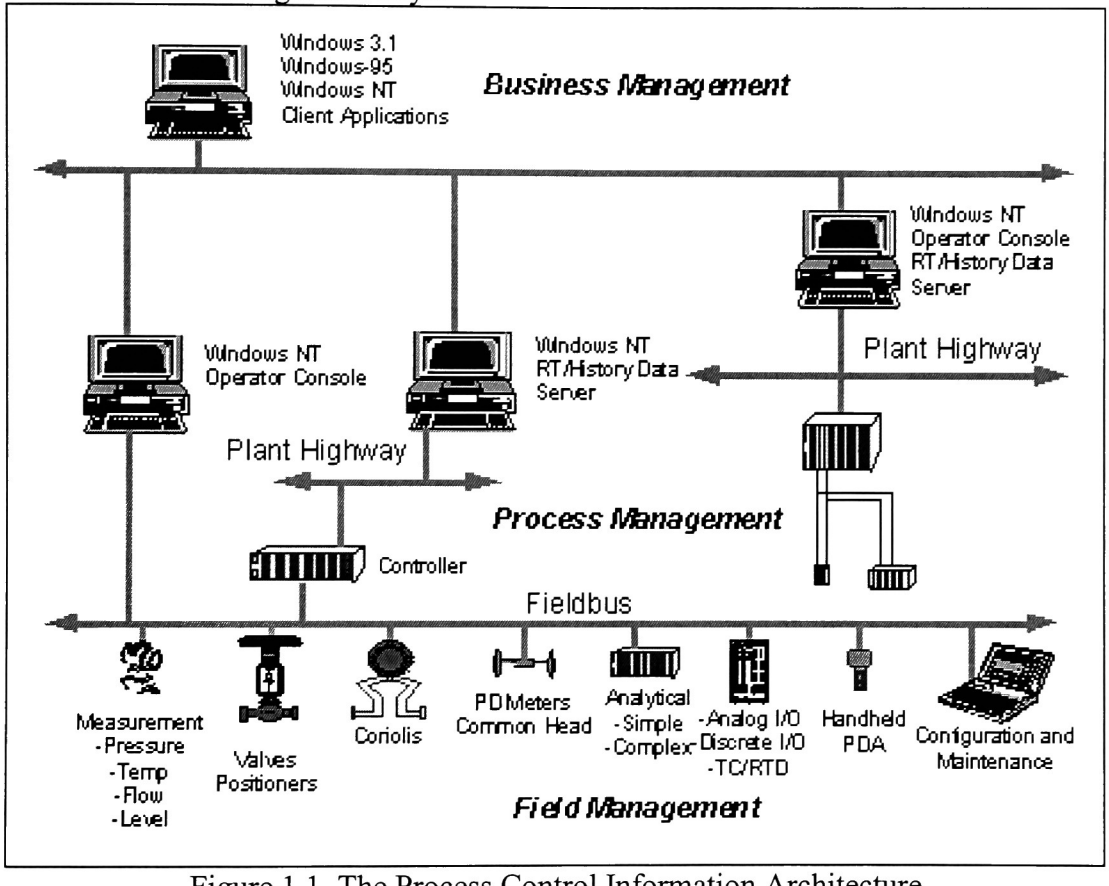


Figure 1.1, The Process Control Information Architecture

(http://www.iconics.com/support/pdfs/opc_specs/opcda20.pdf)

Field Management: This level can also be called the sensor-actuator level. In this level we find several hardware devices that connect to the process being controlled. Figure 1.1 shows various devices such as analog and digital I/O, valve actuators, sensors, etc. in the Field level of the control architecture. With the advent of "smart" field devices, a wealth of information can be provided regarding field devices that were not previously available.

These field devices play a vital role in gathering process related data and providing the same to the next (higher) layer of the architecture. The control algorithm residing in the higher layer processes this data and the necessary control action is initiated, so that the various actuators and positioners residing in the Field Management layer could actuate the controlled variable in order to bring the output variable to the required setpoint.

Process Management: This level is also called the field level by proprietary networks such as PROFIBUS.[2] In this level, the installation of Distributed Control Systems (DCS), Supervisory Control and Data Acquisition (SCADA) systems, and Programmable Logic Controller (PLC) systems are used to monitor and control manufacturing processes. This makes data available electronically which was previously gathered manually.

These systems acquire inputs from the various field devices residing in the lower level of the architecture and then process these inputs by comparing the input value with the setpoint in their control algorithm. Depending on the value obtained, the system would give an output signal to the actuator in the field to either open or close so as to bring the process under control. The usage of either of the above systems depends on the importance of the process being controlled, the number of inputs and outputs, and the scan and refresh rates required to scan each input, and so on.

Business Management: In this level, enterprise wide integration systems such as Enterprise Resource Planning (ERP) are installed to provide effective knowledge management of existing resources. This is accomplished by integrating the information collected from the process into the business systems managing the financial aspects of the manufacturing process. Providing this information in a consistent manner to client applications minimizes the effort required to provide this integration.

ERP consists a single, integrated software program that runs off a single database so that various departments can easily share process related information with each other.[1]

There are several Data Acquisition Networks (fieldbus technology) available in the industry that follow the ISO-OSI reference model and the IEEE Project 802 model for networking. Even though various network vendors (e.g., PROFIBUS, DEVICENET, INTERBUS, ETHERNET I/P, etc.) claim that their network is an "open" system that allows interoperability of devices between networks, but due to differences in protocols followed by these networks in the physical and the data-link layer, devices available today are greatly polarized towards supporting the network that is widely followed in manufacturing processes. This problem prevents a customer from having a world class manufacturing system consisting of a mixture of world-class devices for data acquisition and control.

Due to the differences in protocols followed by various networks, manufacturers are forced to write drivers for their device to be compatible with every network available in the market today, requiring a unique, device-specific chunk of code for the device or network it wants to connect to. Thus the time and resources required by device manufacturers to develop newer and smarter devices are now directed to developing drivers for each individual network standard now available in the market.

In the present situation customers would have to develop client applications that access data from data sources or devices by developing "drivers" for their own package. This leads to the following problems:

- Everyone must write a driver for a particular vendor's hardware
- Hardware features are not supported by all driver developers
- A change in the hardware's capabilities may break some drivers
- Two packages generally cannot access the same device simultaneously since each of them contains independent drivers

An answer to the presented problem would be to develop a universal driver that could be used by all clients connected to the network (making it network and device independent).[1]

The solution to the "driver" problem would be to find a standard mechanism for applications to access data from numerous data sources, either devices on the factory

floor, or a database in a control room. This is the main motivation behind the development of the OLE for Process Control (OPC) Specifications. OPC draws a line between hardware providers and software developers by a mechanism that provides data from a data source and communicates the data to any client application in a standard way. Providing the server (data source) with an OPC interface allows clients to access their devices.

OPC is based on Microsoft's OLE/COM (Component Object Model) technology that provides a set of interfaces for truly open software application interoperability between automation/control applications, field systems/devices, and business/office applications.[3] OPC is based on the client-server paradigm, where the field device (which is the source for data) acts as the server for the various client applications that makes use of the data from the server. The extension of COM is Distributed Component Object Model (DCOM), which extends the client-server communication across machine boundaries. This means that OPC clients and servers may communicate while each runs on a separate computer, provided that each computer is connected within a Local Area Network (LAN). The LAN may include a network server or a simple point-to-point connection between machines.[4]

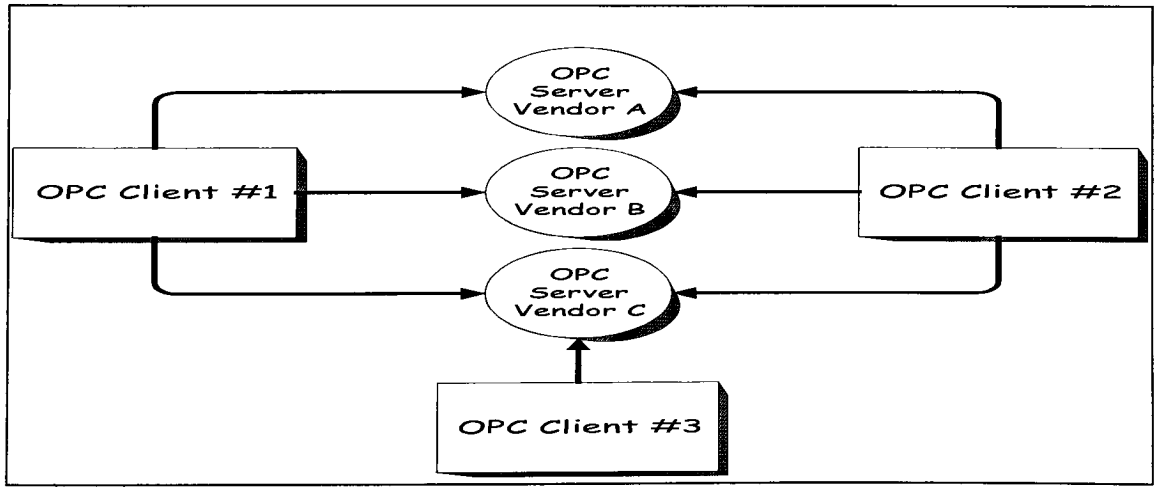


Figure 1.2, Client/Server Relationship (Source-OPC Foundation)

The architecture and design makes it possible to construct an OPC server, which allows a client application to access data from many OPC servers provided by many different

OPC vendors running on different nodes on the network, and a single OPC server can be accessed by more than one client at the same time, as shown in Figure 1.2.

The objective of this thesis is to explore the OLE for Process Control (OPC) technology in depth by highlighting its need in industry, and also to use this technology in an application that is used to illustrate the ability of OPC to integrate dissimilar systems in order to bring together different "islands of automation" existing in today's manufacturing scenario.[5]

The second chapter covers a number of field devices available, which are used to acquire data or information from the field or the process in which it is placed. The third chapter gives an overview of various Data Acquisition Networks available in the industry.

A brief introduction to LabVIEW is made in Chapter 4 in order to give the reader insight into one of the most widely used data acquisition software in the industry today. LabVIEW is a graphical programming development environment based on the G programming language for data acquisition and control, data analysis, and data presentation.[6]

DataSocket Transfer Protocol (DSTP), a technology that is resident in LabVIEW that facilitates data sharing between applications in a computer or different clients in a network is presented in Chapter 5. This technology is presented to highlight the client-server model followed by LabVIEW in order to share information between applications. This technology would be used in tandem with OPC technology and will be presented in the application later.

Chapters 6 and 7 explore the OLE for Process Control (OPC) technology in detail and bring to light a number of case studies or applications that heavily rely on the OPC technology for systems integration. The application of OPC with DataSocket technology to integrate Siemens Simatic PLC with LabVIEW is presented in Chapter 7.

References:

1. OPC Task Force. (1998). "OPC Overview, Version 1.0". OPC Foundation < http://www.iconics.com/support/pdfs/opc_specs/opcda20.pdf>
2. Profibus International. "PROFIBUS AND PROFINET Technology Overview" 2001 <http://www.profibus.com/profibus.html>
3. Barillère R., Baggiolini V., Beharell M., Chmielewski D., Gras P., Milcent H., Kostro K. (1999). "Results of the OPC Evaluation done within JCOP for the Control of the LHC Experiments". Proceedings 1999, International Conference on Accelerator and Large Experimental Physics Control Systems, Trieste, Italy.
4. Janke, M. (2000). "OPC-plug and play integration to legacy systems". Proceedings 2000 Pulp and Paper Industry Technical Conference. Page(s): 68 –72
5. Shimanuki, Y. (1999). "OLE for process control (OPC) for new industrial automation systems". Proceedings. 1999 IEEE International Conference. Page(s): 1048 -1050 vol.6
6. National Instruments Corporation. "Draw your own Solutions-LabVIEW" 2002 <http://amp.ni.com/niwc/labview/what.jsp?node=1381>

2. Sensors Review

The use of Data Acquisition Systems in the real world has exploded in the last few years with the advent of smart sensors and related technology. Here in this chapter we will investigate some of the sensors that are used in applications, in order to acquire data and use the data for control and supervision purposes.

Some of the applications discussed here are:

1. Home Security Systems using Data Acquisition
2. Gas Detection Sensors
3. Global Positioning Systems
4. Advanced pH Measurements
5. Wireless Data Acquisition
6. Biometrics and Intelligent Peripherals
7. Fiber Optic Sensor Technology

2.1 Home Security Systems using Data Acquisition:

Data acquisition systems used in home security are not a new application, but home security is a field or application where sensors are relied on rather heavily, usually beginning with burglar protection and moving into fire protection.

2.1.1 Detectors:

Alarms triggered by photoelectric detectors are simply activated by an intruder breaking an invisible infrared light beam that is passed from a transmitter to a receiver. They can be designed for indoor or outdoor purposes. Magnetic Sensors are used to detect whenever a window or door opened. They are usually mounted inside a door or window frame.[1]

Ultrasonic detectors pick up high frequency (above 20 kHz) sound waves. Typical applications use 9 to 16 Volts DC and 25mA, covering about 400 sqft. Units may be mounted in close proximity to each other and use an alarm with normally closed contacts.

Passive InfraRed (PIR) Detectors can identify a change in energy when a body of one temperature passes through an area of another temperature. PIR detectors can cover an area of 25000 sq. ft. PIR detectors can be combined with a microwave sensor to have a processor analyze their sensed data in order to distinguish people from pets and other false alarms.[2]

Glass breakage sensors use a microphone to receive audio signals. An alarm is based on the signal meeting a specific frequency, signature, and timing after it is run through a microprocessor. Applications require that the sensor be lined up with the centerline of the window and be within approximately ten feet of the window. The detectors may be used with plate, tempered, laminated, and wired-types of glass.[3]

Floor-pad sensors sense an unwanted presence moving across the pad and send a signal to the processing unit.

2.1.2 Fire Alarms:

There are three types of fire alarms that are common in today's home. First, the older ionization detector senses invisible smoke particles. Next is the photoelectric detector, which responds to visible smoke particles. Third is the heat detector, which is often found in kitchens where grease buildup can render the other two units ineffective.

Once a fire detector sends a signal to the digital communicator in the home's console, the digital communicator places a call to a central station, where a computer sends the alarm information to a live dispatcher. The dispatcher then verifies the alarm by calling the home where the alarm originated. If there is no one home or the dispatcher is given an incorrect clearance code, the dispatcher notifies the pre-arranged fire or police department.[4]

2.1.3 Wire versus Wireless Home Security:

Wireless home security is considered by many to be the way of the future. Although there are a few advantages of wireless types of home security systems over the conventional hard wired units, there are many more disadvantages.

Advantages of Wireless Home Security:

- Quick and simple to install
- No worry of wires breaking or being damaged by rodents
- No risk of wires carrying lightning

Disadvantages of Wireless Home Security:

- Battery failure can mean no security
- An intruder can tamper with battery
- Large objects or mirrors can interfere with the radio signal
- Other wireless devices can interfere with the signal on which the wireless device depends on
- For every wireless detector that is added to the system, the odds of failure increase

Although wireless systems are highly reliable, hard-wired systems are still the most trouble-free and reliable. Hence, we conclude that sensors play a very important role in Home Security Systems and play a positive role in safeguarding one's valuable property.[5]

2.2 Gas Detection Sensors: [6]

Sensors in gas production are crucial because of the volatility of the process. A ***Catalytic*** sensor is an example of a gas detection sensor; they are used in specific areas of the process to detect leakages. The heart of a catalytic sensor consists of a pair of computer-matched pellistors (bead) that provides ideal electrical resistance in clean air. When flammable gas is present, the active bead catalyzes the combustible gas molecules; while the inactive (reference) bead balances the reaction and compensates for the normal changes in ambient environment. The instrument measures the resulting increase in

resistance and translates it into percent of lower explosive limit (LEL). The catalytic sensors provide excellent sensitivity, response time, and resistance to physical shock and vibration. The key in gas detection is placement and reaction time. There are specific locations that pose the greatest chance of a leak that is where they need to be placed.

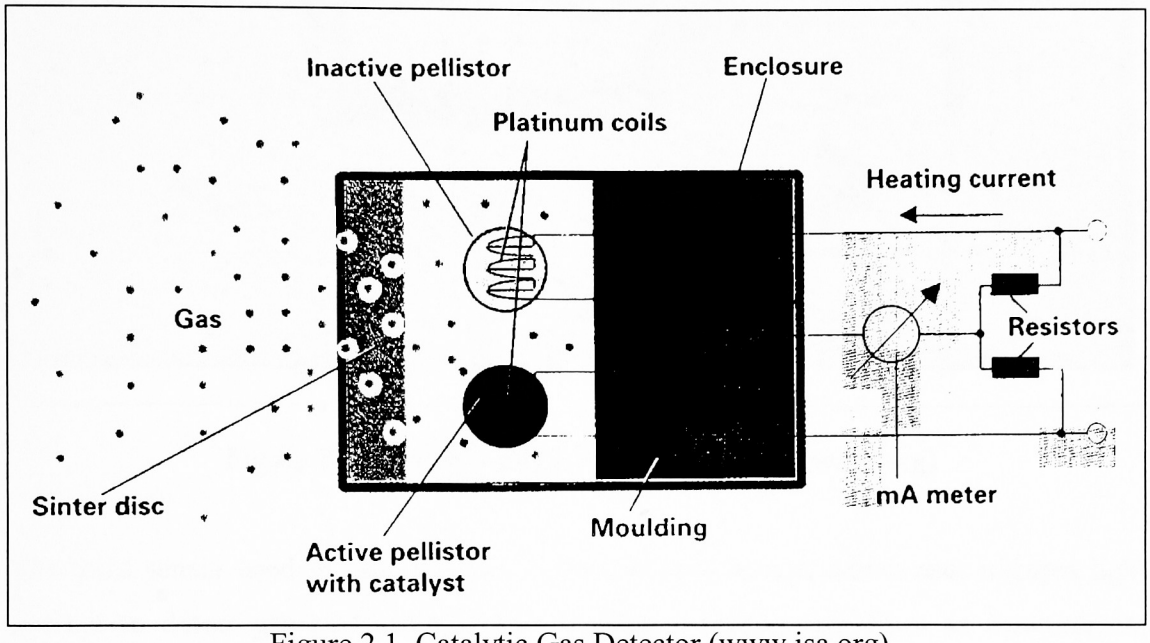


Figure 2.1, Catalytic Gas Detector (www.isa.org)

The second sensor that would detect a gas leak is the ***Electrochemical*** sensor, which uses an electrolyte and an embedded NTC sensor to detect gas. Electrochemical gas sensors contain various components designed to react with a specific toxic gas; the reaction generates a current, which is measured by the instrument and translated into a concentration value (PPM or PPB). Gas diffusing into the electrochemical sensor reacts at sensing electrode to cause current to flow. A more precise term for this type is the "porous-electrode amperometric gas sensor". It is also called a fuel cell sensor. It will respond to gases that can be electrolytically reduced or oxidized on a metallic catalyst such as platinum or gold. Gases measured are O_2, CO, NO_2, NO, H_2S, SO_2, NH_3, HCl, HCN, Cl_2 and organic vapors such as alcohols, aldehydes, or ketones. Electrochemical have an advantage of being fairly inexpensive compared to others.

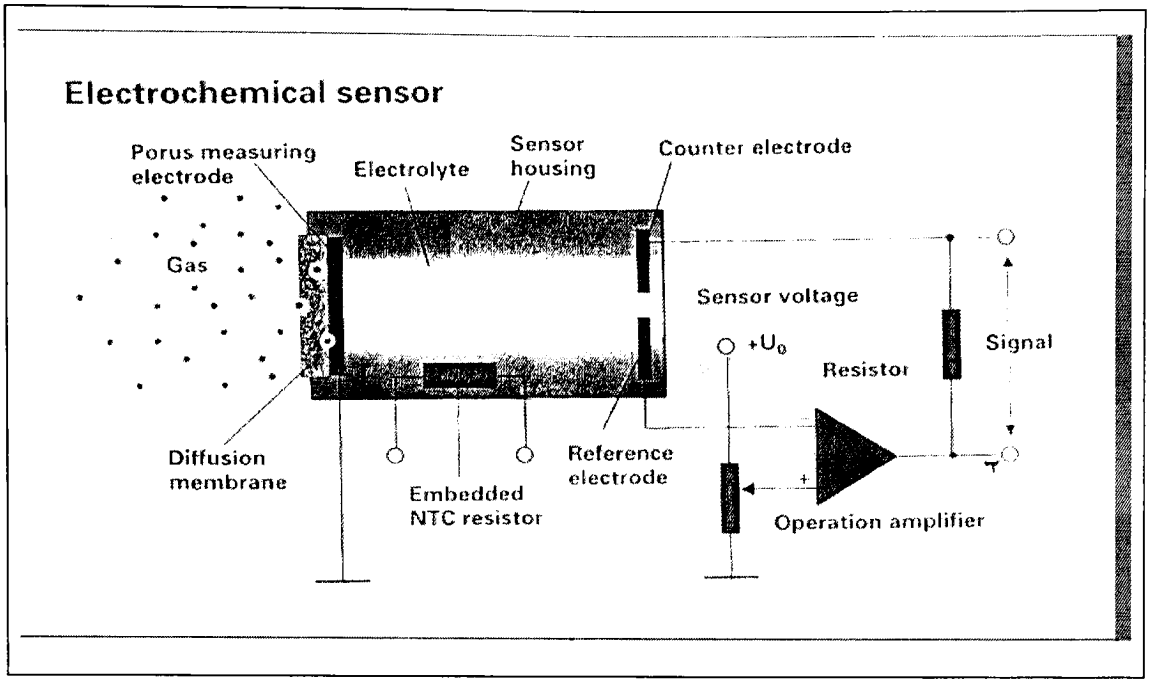


Figure 2.2, Electrochemical Gas Detector (www.isa.org)

The third sensor used in gas detection is the ***Infrared*** sensor, which uses infrared light sources to detect gas molecules. Infrared radiation propagating from an incandescent emitter passes through a volume of gas, which is reflected by a mirror onto two fixed detectors. One detector is tuned to a wavelength that is absorbed by hydrocarbons or CO_2, while the other detector is tuned to a nearby wavelength that is not absorbed. The ratio of the output voltages of the two detectors is used to detect the concentration of gas. As gas concentration increases, the incremental output produced by the detectors decreases. The instrument then uses a gas specific algorithm to linearize the signal.

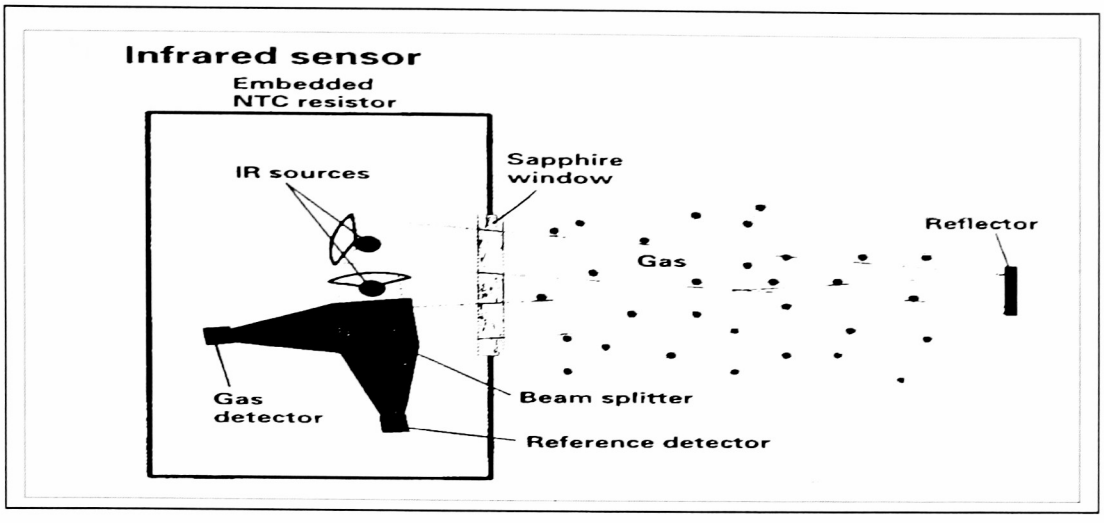


Figure 2.3, Infrared Sensor (www.isa.org)

2.3 Global Positioning Systems (GPS): [7]

Global Positioning Systems are rapidly becoming a large part of our everyday lives. GPS is a space-based radio positioning and time transfer system, which was conceived and operated by the U.S. Department of Defense (DOD) and is still funded and controlled by them. The DOD has a nominal GPS operational constellation that consists of 24 satellites that orbit the earth in 12 hours. There are often more than 24 operational satellites in orbit, with the DOD launching new satellites to replace the old ones. The D.O.D has split the sky into 6 orbital planes with at least 4 satellites in each plane. This 'constellation' provides a GPS user with between 5 and 8 visible satellites from any point on earth.

The GPS also consists of a system of tracking stations around the world. The master control facility is located at Schriever Air Force Base in Colorado. The monitor signals measure signals into position, velocity, and time estimates. Three satellites are needed to compute position in 2D and four are need to compute position in 3D. GPS would be able to compute a 3D position with only three satellites if the receiver possessed a perfect clock, which is rarely possible.

Originally, the D.O.D instituted Selective Availability to degrade the GPS signal for civil users. This was necessary to ensure that adversaries do not exploit highly accurate GPS signals and use them against the US and its allies. With the elimination of Selective

Availability on May 1st, 2000, civil users no longer experience the 100 meters random errors that were an ordinary event in the past. Today GPS still has several possible errors for e.g., noise, and bias. Noise error occurs because of a combination of code noise and noise within the receiver. Noise error can account for 2 meters of error. Bias errors occur because of Selective Availability, uncorrected satellite clock errors, delays in the atmosphere, or reflected signals. Bias errors can account for up to 12 meters of error. Blunders are computer or human errors in the system.

Bias errors have been corrected with the invention of the Differential Global Positioning System. The idea behind this is to correct bias errors at one location with measured bias errors at a known position. The accuracy achieved with DGPS is less than a meter.

2.3.1 Differential Global Positioning System (DGPS): [8]

DGPS works by placing a high-performance GPS receiver (reference station) at a known location. Since the receiver knows its exact location, it can determine the errors in the satellite signals. It does this by measuring the ranges to each satellite using the signals received and comparing these measured ranges to the actual ranges calculated from its known position. The difference between the measured and calculated range is the total error. The error data for each tracked satellite is formatted into a correction message and transmitted to GPS users. The correction message format follows the standard established by the Radio Technical Commission for Maritime Services, Special Committee 104 (RTCM-SC104). These differential corrections are then applied to the GPS calculations, thus removing most of the satellite signal error and improving accuracy. The level of accuracy obtained is a function of the GPS receiver. Sophisticated receivers like the Starlink DNAV-212 and INVICTA 210 series can achieve accuracy on the order of 1 meter or less.

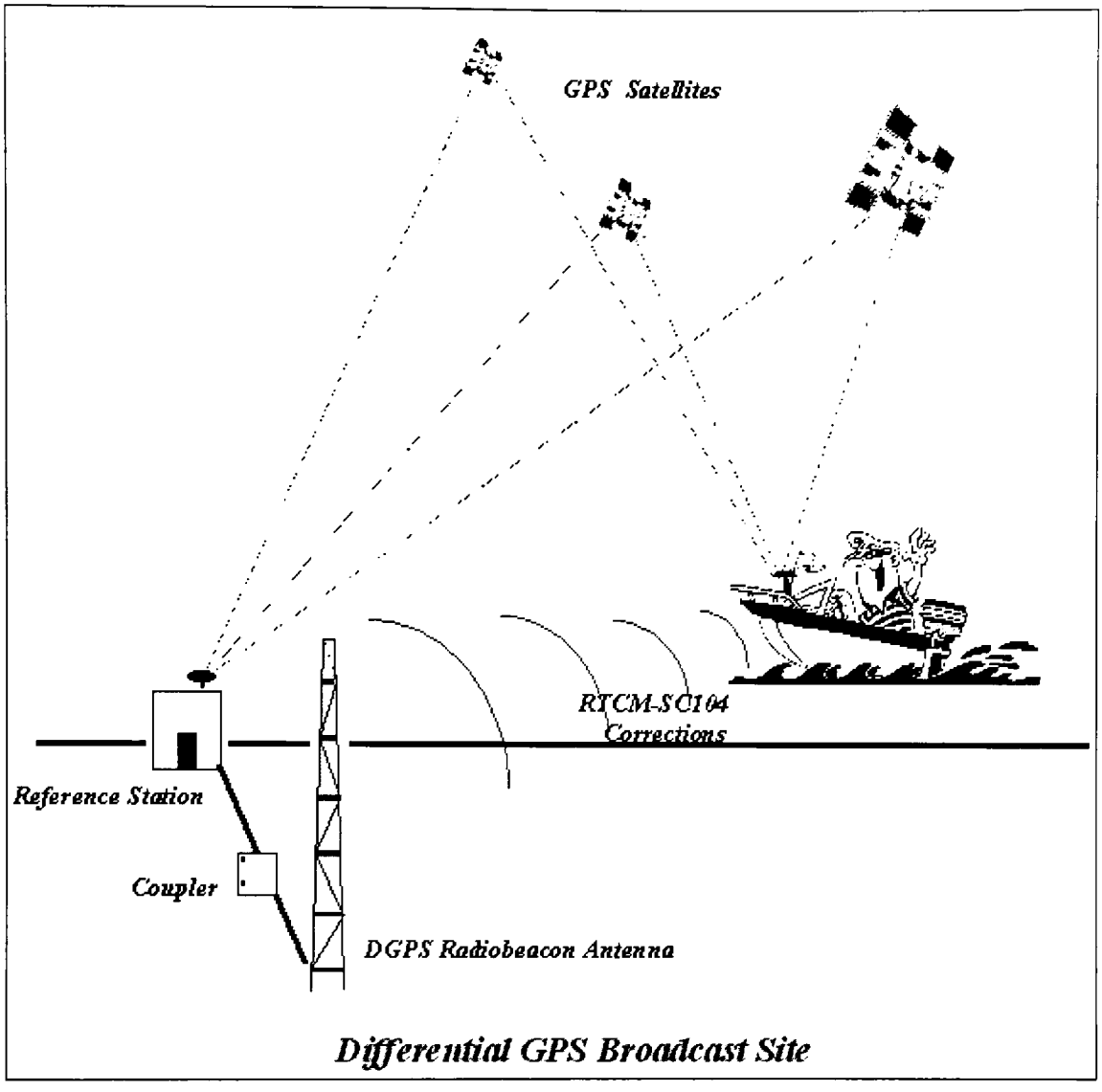


Figure 2.4, GPS Components (www.starlinkdgps.com)

2.3.1.1 Reference Station:

The reference station GPS receiver knows exactly the position of its antenna; therefore it knows what each satellite range measurement should be. It measures the ranges to each satellite using the received signals just as if it was going to calculate position. The measured ranges are subtracted from the known ranges and the result is range error. The range error values for each satellite are formatted into messages in the RTCM SC104 format and transmitted continuously.

2.3.1.2 Modulator:

Depending on the transmission format, the modulator encodes the data as necessary for transmission. For example, in the free USCG system, the modulator creates a carrier signal, which varies using MSK modulation. A "one" data bit is represented by a advancing carrier phase and a "zero" bit by a retarding carrier phase. The modulated carrier output from the modulator is connected to the transmitter.

2.3.1.3 Transmitter:

The transmitter is basically a power amplifier, which is connected to an antenna system. The modulated carrier is amplified and driven to the antenna. In the USCG system, the transmitter is 250-1000 Watts and operates in the 300KHz frequency range. The amplified signal is radiated via the antenna to remote DGPS receivers for real-time position correction.

2.3.1.4 DGPS Correction Receiver:

A DGPS correction receiver decodes the signals received from a reference site. Data are formatted into a serial RTCM SC104 data stream and provided to the remote GPS receiver. There are many types of DGPS correction receivers.

2.3.1.5 GPS Receiver:

The GPS receiver measures ranges to each satellite, but before the measurements are used to calculate position, corrections received from the DGPS receiver are applied to the measurements. The position is then calculated using the corrected range measurements providing vastly increased accuracy.

2.3.2 GPS Application (Automobile): [9]

One of the many models of DGPS that are readily available in the automobile market is the RGP 2202 model made by RoyalTek.

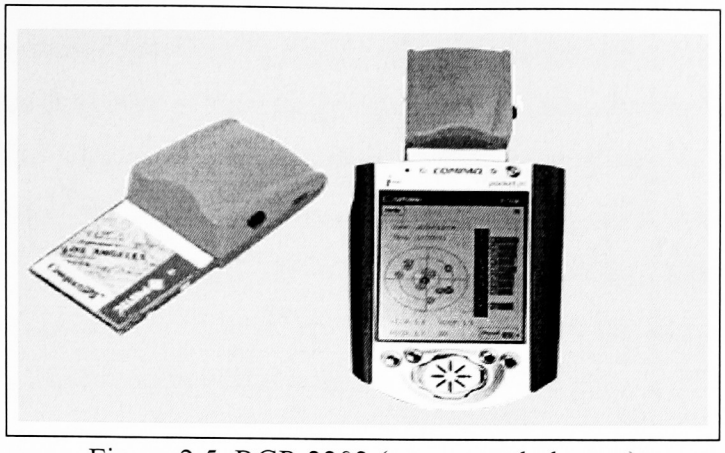


Figure 2.5, RGP-2202 (www.royaltek.com)

The system consists of 4 components. Three of the components of the RoyalTek model are satellites in orbit, while the other component is the module that mounts on your dashboard. The system works on the basis of triangulation of 3 satellites in orbit to calculate ones location on ground. It operates under the same principle as discussed above.

2.4 Advanced pH Measurements: [10]

Potentiometry, or the study of pH measurement began humbly, in the food industry, as a substitute of manually testing food. In the past, in order to determine the quality of a food product, a person would have to physically taste the food. While many times this would be a pleasurable task, such as with candy or wine, if the wine had gone bad, the taster would be left with the unpleasantness of vinegar in his or her mouth, or in the worst case would be killed by the ingestion of poisonous food. When potentiometry was developed, this allowed the testing of food through means of pH measurement, the determination of relative acidity.

This determination is based on the presence of the Hydronium ion, or H_3O. If the Hydronium ion is present at a concentration higher than 0.0000001 mol/L (10^{-7} mol/L), the solution is considered acidic, and if it is below this value it is considered basic. The determination of relative acidity or basidity is based on the offset from this neutral value.

The term pH was coined by a Danish biochemist named Soren Paul Lauritz Sorensen. It stands for 'pondus Hydrodenii' (hydrogen exponent). In order to simplify and to create a governing scale, he created a ph scale. The scale is between 0 and 14, and is simply the negative logarithm of the Hydronium ion concentration. Therefore, if the Hydronium ion concentration in a substance were 10^{-7} mol/L, its pH would be 7, the negative of the exponential value. Like wise, if the Hydronium ion concentration was 10^{-5} mol/L, the ph value would be 5, slightly more acidic than the neutral value of 7.

While measurement of pH originated to solve problems associated with the food quality assessment, it has since stemmed to many other useful applications. It is used to measure everything from corrosion and scale formation in boiler and cooling towers to waste treatment. It is still used in the food industry for quality control, and in the pharmaceutical and biotechnology industries to enhance the growth and production of useful organisms.

2.4.1 Basic Principle of Operation: [11]

pH is a measurement of a hydrogen ion activity of a solution

$$pH = - \log [H+]$$

Most pH sensors use a glass membrane electrode, which develops an electric potential. This potential varies in proportion to the hydrogen ion activity. This glass membrane is only sensitive to hydrogen ions.

A reference electrode completes the circuit between the glass electrode and the solution being measured. It measures the electrical potential generated across the glass electrode and transmits the electrical potential to the monitor. In the monitor, this potential is amplified by the signal-conditioning module. The potential of the sensor varies between sensors and changes with age. This is why periodic calibration is necessary. The calibration of a pH sensor is also dependant on temperature. For this reason, most pH sensors include automatic temperature compensation.

2.4.2 Present Technology: [12]

Today, instead of two probes, a combined pH electrode is used, and is contained within a single probe. This process continues to improve as well, as originally glass electrodes were used. Glass electrodes diminish overall performance, and increase the overall cost. There are currently products in the market that have greatly improved the pH measurement process.

2.4.2.1 Durafet II pH Electrodes: [12]

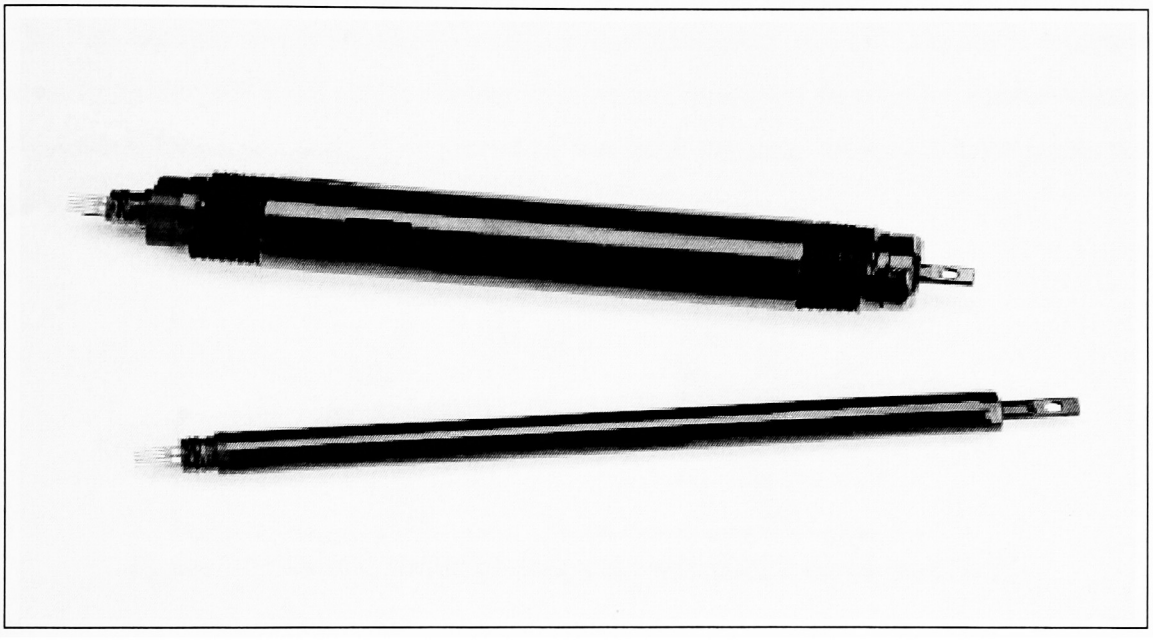


Figure 2.6, Durafet II pH Electrode (www.content.honeywell.com)

The Durafet II pH Electrode by Honeywell is the next generation pH electrode with the technology and performance to meet tomorrow's pH challenges today. It provides superior stability, speed, and durability to improve product quality and reduction of life cycle costs. Durafet II pH electrodes may successfully replace glass electrodes in many established pH applications and deliver better performance due to its superior technology.

2.4.3 Technology of the Durafet II pH Electrode:

The virtually unbreakable Durafet II pH Electrode incorporates a solid-state, ion-sensitive field-effect transistor (ISFET) technology. This technology delivers increased system stability and improved electrode performance and increases electrode life.

2.4.3.1 ISFET Technology: [13]

The ISFET (Ion Selective Field Effect Transistor) methodology for ion measurement is developed on the basis of the MOSFET (Metal Oxide Silicon Field Effect Transistor). The first attempts to use the ISFET as pH sensor were made in 1970. The basic principle underlying the MOSFET is the control of a current flowing between two semiconductor electrodes. Drain and source are placed on the same element, with a third electrode, the gate, between them.

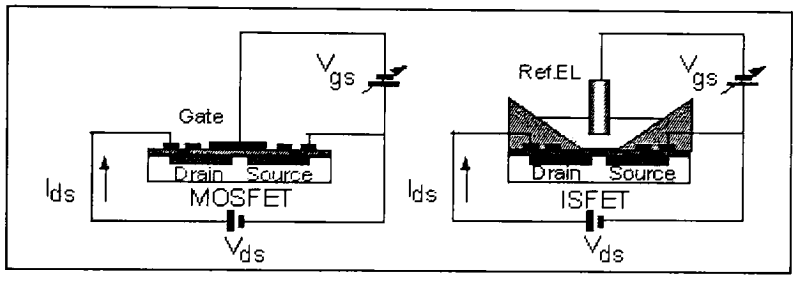


Figure 2.7, Normal MOSFET compared with ISFET (www.sentron.nl)

The metallic gate electrode is insulated against drain and source by means of silicon dioxide and can only influence the drain-source current electrostatically. The extremely high input resistance of the gate electrode means that no large input power is required to control this current.

The MOSFET's gate consists of a metallic coating and is used as an electrode to control the drain-source current through the external potential Vgs. In the case of the ISFET the metallic gate is replaced by a special oxide-coated gate, which is sensitive to hydrogen ion concentration. When immersed in a liquid, the electrical circuit Vgs is closed with the reference electrode and the hydrogen ion concentration in solution can influence the

drain-source current. The fundamental ISFET pH measuring system is shown in the following figure.

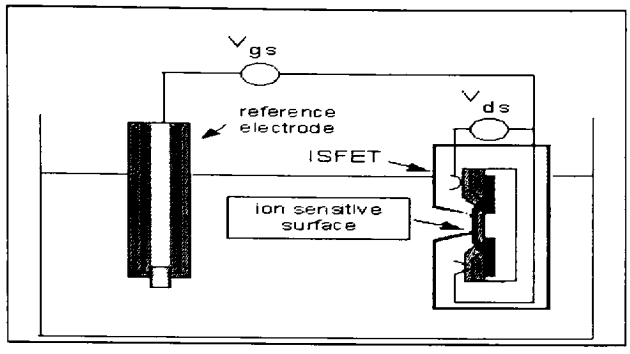


Figure 2.8, Schematic of an ISFET pH measuring system (www.sentron.nl)

2.5 Wireless Data Acquisition: [14]

In many manufacturing applications, the installation of cabling is costly, problematic, or impossible. With state-of-the-art Radio Frequency (RF) technology it is possible to implement a wireless link to an input/output system. It is apparent that the Data Acquisition systems are beginning to see considerable benefits as wireless and networked systems replace costly and inflexible wired installations.

Flexibility and low cost are the two major advantages of using radio-transmitted data. Rather than collecting data on location and transcribing it by time consuming and error prone human recording, a system can be set up where data is collected on a PC that is in a convenient, non-hazardous location. Channels can report data or events as they happen, to a computer serving a plant or refinery. The ability to monitor trends via wireless communication in real time makes it possible to observe a problem as it develops and take immediate remedial action on it, where as before this observation may have been impossible or costly. Wireless transmitting and receiving units are quite compact and can be adapted to fit into the tightest spaces in both permanent and temporary installations.

Wireless DAQ generally falls into one of two categories. The first category is remote data collection where normal wiring techniques are not possible and data acquisition may not be feasible. In these cases, the area being examined might be miles away from the plant.

Moving to a wireless data acquisition system to monitor data may save a company in terms of travel time and labor. Remote wireless data collection is of interest to companies who monitor monthly usage levels at different residences, need to respond to emergency situations such as gas leaks, or need to continuously monitor the status of an area.

The second area of wireless data acquisition is plant or warehouse monitoring. In this case there would be a wireless network, which is a group of sensors linked by a wireless medium to perform distributed sensing tasks and transferring data to the main control panel in the factory control room.

2.5.1 Wireless Data Acquisition with MiniDAT: [15, 16]

MiniDAT enables you to monitor, acquire and process data from remote sensors over wireless links. Combining wireless LAN technology with National Instruments' extensive DAQ capabilities, MiniDAT is the latest step in the evolution of wireless data acquisition. High-speed wireless data acquisition saves significant time and money addressing applications where it is difficult, expensive or maybe impossible to install a hard-wired DAQ system. ViaSat's MiniDAT combines wireless local area network technology with National Instruments' extensive DAQ capabilities to offer the state of the art in wireless data acquisition at speeds up to 200K 16-bit samples/second

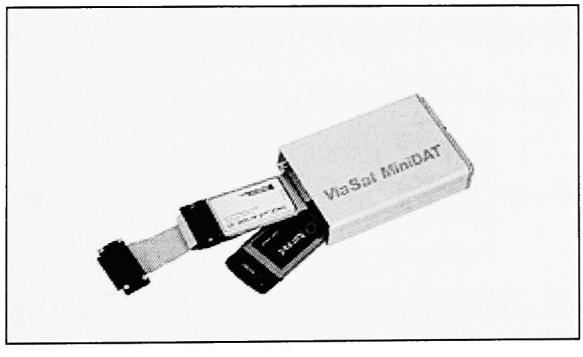


Figure 2.9, MiniDAT (www.viasat.com)

MiniDAT provides:

- High-speed, multi-channel DAQ anywhere, anytime, inside or outside without wires at speeds up to 200K 16-bit Samples/second

- Wireless DAQ which is fast and easy to install, eliminating the need to run cabling, which may be difficult or impossible
- An Ethernet-based DAQ network that can acquire data at up to 11 Mbps, a data rate crucial to a growing number of applications.
- Wireless DAQ systems that cost less to own and maintain than hard-wired DAQ. Gone are costly cable repairs. Less time and money are spent when setups are reconfigured or expanded.
- A high-speed, multi-channel, networked wireless DAQ for National Instruments users
- Wireless DAQ that can be attached to TCP/IP networks, including the Internet, making remote access possible

Hard-wired computer-based data acquisition is characterized by:

- Sensors and transducers generate electrical signals
- Most transducer outputs must be conditioned to provide signals suitable for the data acquisition subsystem, DAQ
- The DAQ converts electrical signals into digital data
- Output of the DAQ is wired to a computer

Application software, quite often developed using National Instruments' LabVIEW, is used to display the data being acquired and/or save it to the computer's disk for post-processing.

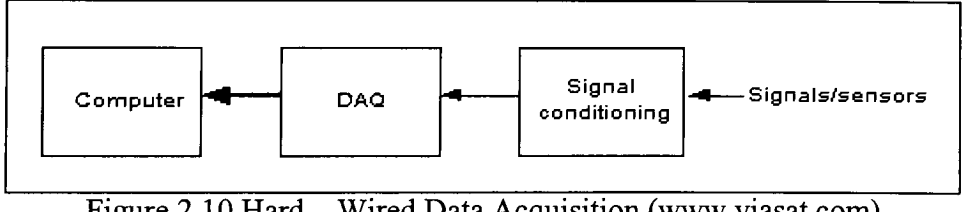


Figure 2.10 Hard – Wired Data Acquisition (www.viasat.com)

There is a definite need for wireless DAQ since:

- In many industrial applications, it is difficult, expensive or maybe impossible to connect the DAQ to a computer using cables
- Long signal wires can introduce noise and corrupt analog signals

- Wiring is an obvious problem when data are being acquired from a moving source

The first generation of wireless DAQ used telemetry. Data being acquired is first serialized into a stream of ones and zeros and this data stream is used to modulate a carrier frequency. At the receiving end, the stream of bits must be separated from the carrier and converted into intelligent data.

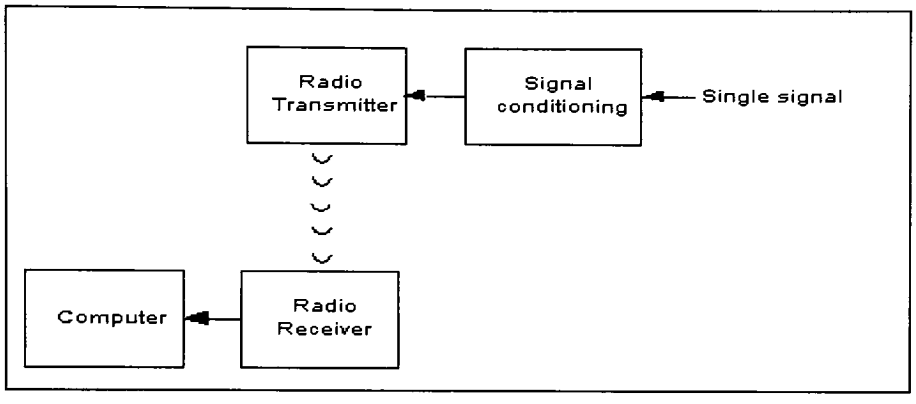


Figure 2.11, Data Acquisition by Radio (www.viasat.com)

Telemetered DAQ is slow, sends data in only one direction and is single channel. To instrument five signals, five transmitters and receivers will be needed. The latest approach to wireless DAQ is to utilize industry standards such as PCMCIA, TCP/IP and IEEE 802.11 Wireless Local Area Network, WLAN, and incorporate all of the individual DAQ units into a wireless local area network.

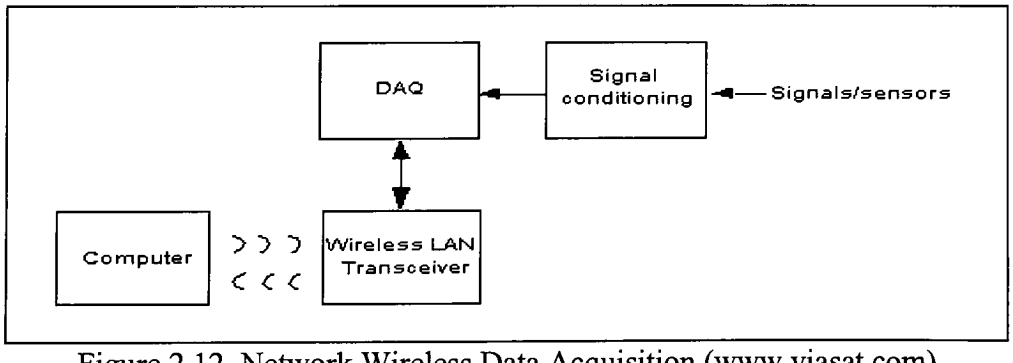


Figure 2.12, Network Wireless Data Acquisition (www.viasat.com)

In the above configuration, data are acquired by the DAQ subsystem, which transfers the data to a WLAN transceiver. Data transfer between the DAQ subsystem and the

computer uses TCP/IP exactly the same as any other Ethernet connection. Data can flow in both directions, offering the option of remotely controlling the DAQ.

The following configuration is familiar to many NI users: a SCXI chassis is connected to a DAQCard and the DAQCard is plugged into a laptop running a Virtual Instrument (VI) created using LabVIEW.

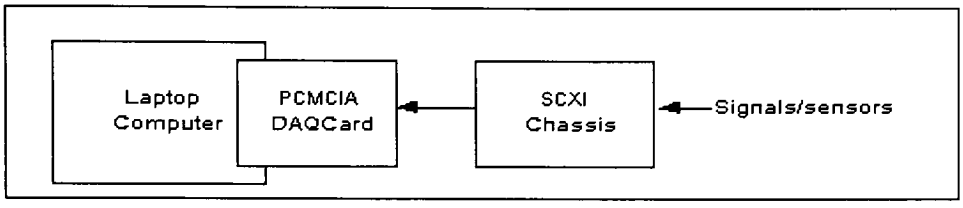


Figure 2.13, NI Portable Data Acquisition (www.viasat.com)

MiniDAT provides high-speed, multi-channel, networked DAQ tightly integrated with National Instruments.

To make the above NI data acquisition wireless, perform the following tasks:

1. Disconnect the SCXI chassis from the DAQCard
2. Remove the DAQCard from the laptop
3. Connect the standard NI cable that is connected to a DAQCard inside the MiniDAT to the SCXI chassis
4. Plug the WLAN Card into the laptop
5. Load the WLAN driver software on to the laptop
6. Establish a wireless link between the laptop and the MiniDAT using RDA
7. Assign the MiniDAT a local device number for the Virtual Instrument
8. Run the VI

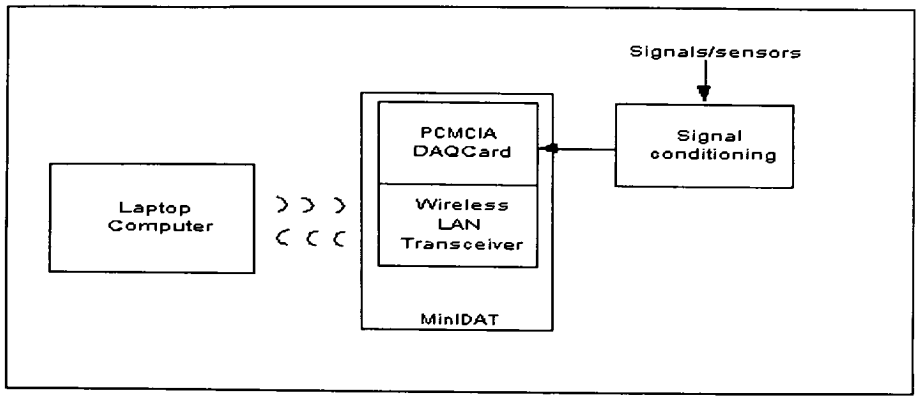


Figure 2.14, Networked Wireless NI Data Acquisition with MiniDAT (www.viasat.com)

Changing to wireless is totally transparent to the user and the Virtual Instrument will now run as it did before. Up to 16 MiniDATs can be networked together and controlled by one computer. Using an access point, this wireless network can be connected to the Internet or any other network, which offers some interesting possibilities. Connecting a WLAN to the Internet via satellite communications can provide Internet access to monitor and control remote data acquisition sites.

2.5.2 Future of Wireless Data Acquisition: [16]

Industry observer Dataquest states, "The efficient collection of data from remote devices is becoming a fundamental requirement for competitiveness for many companies." Lucent Technologies' Bell Labs predicts that the entire world will be encased in a "communications skin" by 2025. This "skin" will consist of millions of electronic measuring devices – thermostats, pressure gauges, pollution detectors, cameras, and microphones – monitoring cities, roadways and the environment. All of these will transmit data directly into the network, just as our skin transmits a constant stream of sensory data to our brains.

2.6 Biometrics and Intelligent Peripherals: [17]

Automatic Identification and Data Capture (AIDC) is the term used to describe data collection by means other than manual notation or keypad input. The most advantageous implication of automatically captured data is a more efficiently run organization. AIDC technologies can be broken down into many categories, one of which is biometrics.

Biometrics is based on the principle that everyone has unique physical characteristics that a computer can be programmed to recognize. It uses mathematical representations of physical characteristics to verify identity. Biometrics is defined by Merriam-Webster dictionary as "the statistical analysis of biological observations and phenomena." [18]

Biometrics can be used both for recognizing people, such as ATM verifying that the person trying to take money from an account is the actual account owner. Biometric advancements are being made in face recognition, voice recognition, fingerprint recognition, lip movement recognition, and iris scan technologies.

2.6.1 General Biometric System: [19]

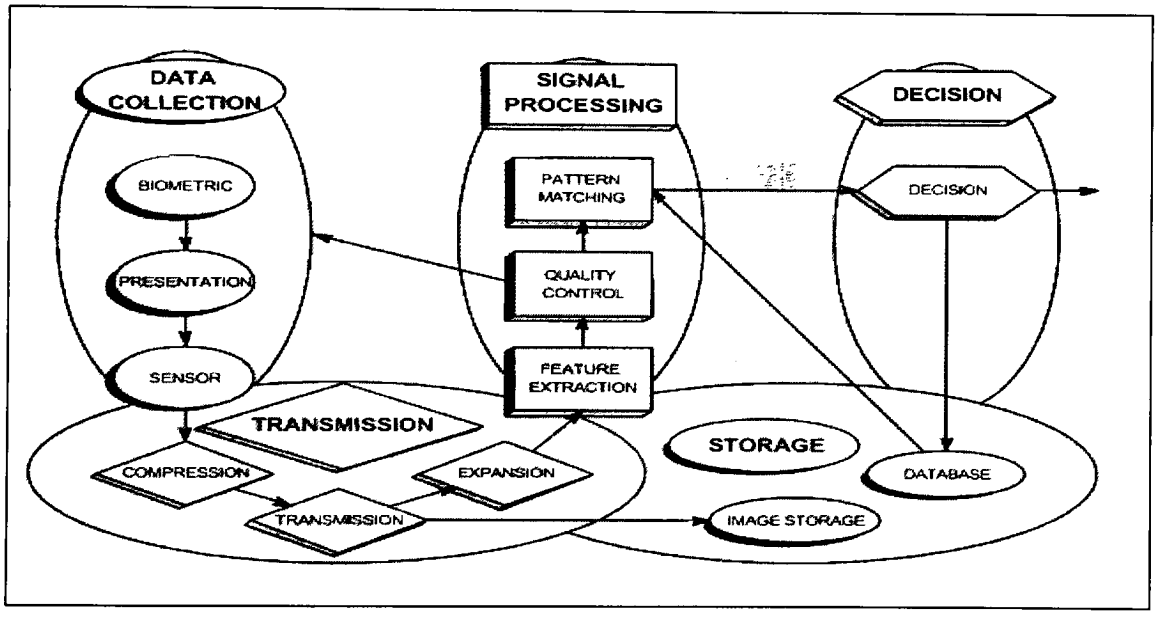


Figure 2.15, Biometric System (Source - Biometrics Working Group)

While biometric applications themselves will be unique, the process followed to get the information is fairly common. The first phase of biometrics is the capturing of information. A sample has to be taken first, in order to compare later on. This sample is usually called as the biometric template. In most instances a number of samples are taken in order to give a more accurate representation. The samples are then averaged together to construct the template.

The second phase of the biometrics is storage. In each instance there are multiple storage options available. They include storage within the biometric device, at a centralized location or within a personal storage card. Each case has its advantages and disadvantages. When the information is stored within the device, the retrieval process is faster because the retrieval of the information does not need any outside processors. However one is limited to the amount of available space in the device. When the information is stored in a central location, you gain the added advantage of being able to store your data within a secured network environment. Also more amount of information can be stored in a centralized location.

The third phase of the biometrics process is the network. The type of network that would be used depends on the application. Some of the higher end biometrics has internal network ability. They often use a RS-485 or RS-422 with a proprietary protocol. This allows a number of devices to be networked together, without the added expense and inconvenience of additional equipment. In other cases the existing network structure may be able to be adapted and integrated with the biometric device.

The final phase of the biometric process is verification or recognition. In a verification system the user must also input some data, be it typed password or a spoken phrase or name. The system will then either allow access based on a positive identification, or not allow access if the name and face do not match. A recognition system will take the measured biometric and attempt to identify the person using the system, thus either allowing or disallowing access to the system. This is where the live biometric of the user is compared to the template that gathered earlier. One aspect of this is setting a parameter that determines the number of attempts a person gets before finally rejecting him. Another aspect is the false rejection rate v/s false alarm rate, and thus let the false rejection rate respond. This is a standard that is developed by the individual company for their individual application.

2.6.2 Biometrics Applications: [20]

One of the key uses for biometrics is in security. The current methods for securing identity include passwords, PINs, and tokens, such as a driver's license or key. However, these are vulnerable to theft, loss, forgery or misplacement. Because biometrics uses biology, it is not vulnerable to any of these. Every person's biology is unique and cannot be reproduced. Security biometrics is applicable in the home, the bank, the office and the medical establishment. If a home security knows that the person climbing through the window is the owner of the house, it will know that the owner simply cannot find the house keys, and the alarm will not go off.

Higher security in a bank is very desirable, both within the bank and at the ATM. Using biometrics at an ATM would eliminate the need to have a card with you and the PIN to be memorized, both things that could be lost or forgotten.

Biometrics is used in today's society for identification and verification purposes; each of these is useful in its own right. It is necessary to determine which scenario is right for your specific application; a wrong decision would decision could end up costing your company thousands of dollars.

2.7 Fiber Optic Sensor Technology: [21]

In recent years the research and development of optical fibers has redefined the limits of glass technology. This development has led to the expansion of various applications for fiber optic sensors.

Optical fiber was developed as an attempt to compete with other transmission media. As a result a number of key characteristics in optical fibers, research and development of optical fibers has flourished. The positive attributes of the composite media of optical fibers has led o the continued efforts to integrate fiber optic systems into the mainstream environment.

Fiber optic cables provide tough competition when compared to copper twisted pair or coaxial cable. In comparison, optical fibers outer diameter (1mm) is smaller when compared to the outer diameter of coaxial cable (10mm). Optical fiber is also one-third to one-tenth lighter than that of conventional communication cables. Other key characteristics that set optical fibers apart from coaxial cables are its good flexibility, broad bandwidth capability, chemical composition that is free from rust, ability to transmit large amounts of information per unit cross-sectional area and its low ratio of information loss. Additional key properties include optical fibers that are free from electromagnetic induction and lightning damage, prevent a limited amount of light leakage or external light entering the fibers to propagate, and its resistance to high temperature. Optical fibers can be utilized in flammable and explosive environments because there is no electrical discharge of particles at the spliced points. This adds a safety factor to the material and other influential characteristic. These characteristics are typical of the silica-based optical fibers and provide outstanding features as compared to the conventional copper twisted pair or coaxial cables. Data transmission that requires rates faster than 100Mbps will need to be transmitted by fiber optics. Fiber optics is the only medium that is able to handle these rates with confidence.

2.7.1 Optical Fiber Sensors: [21]

Intrinsic and extrinsic properties are two basic components to fiber optic sensing. Intrinsic fiber is considered to have a sensor element that is composed of the fiber itself, where as extrinsic sensors are sensor elements that non-fiber elements. The sensor element is the one that detects the properties of external forces (i.e., displacement, pressure, vibration, temperature, radio waves, electromagnetic field, etc.).

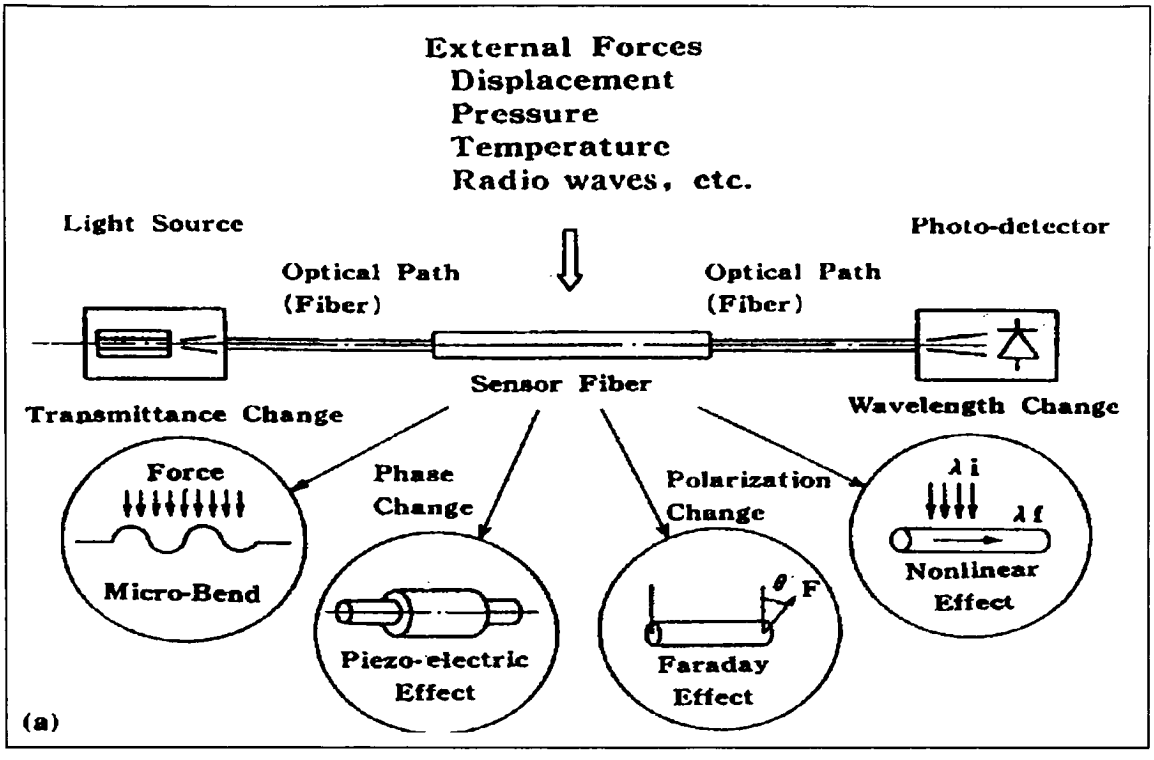


Figure 2.16, Intrinsic Fiber Sensors (Krohn, K. A – Fiber Optics Sensors: Fundamentals and Applications)

In the intrinsic fiber sensors, external forces such as pressure, vibration, temperature, or an electromagnetic field, interact with optical fiber and cause bends, distortion, and a change in the refractive index of the sensing fiber. As a result, external forces influence light wave properties traveling through the fiber. A change in the light wave properties, such as amplitude, light intensity, is detected and facilitates the measurement of the degree of the data interacting with the fiber.

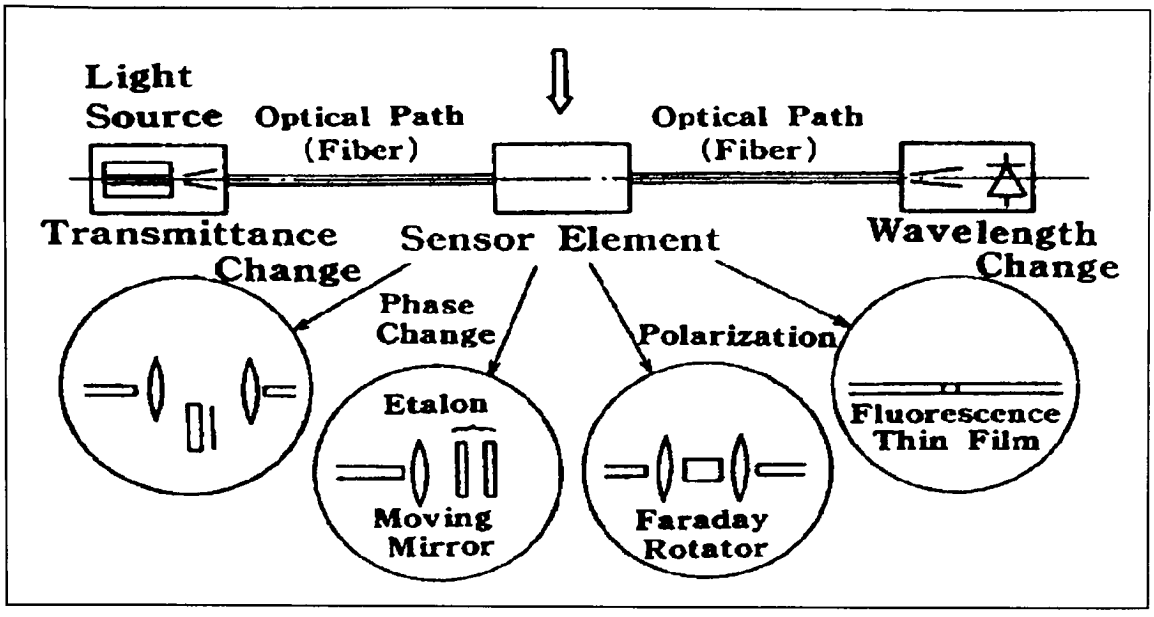


Figure 2.17, Extrinsic Fiber Sensors (Krohn, K. A – Fiber Optics Sensors: Fundamentals and Applications)

In Extrinsic fiber sensors, the crystal that enables the refraction of light into two slightly different directions to form two rays (birefringent crystal), intensity masks, or thin film absorber may be utilized in the form of a sensor element to detect external forces and relay the information back appropriately. The sensor element is incorporated into the optical path, consisting of fibers. The external forces interact with the detection element and modulate the properties of a light wave propagating through an optical path between the light source and the photo-detector. The detection of changes in the optical properties enables the measurement of the magnitude of measurands interacting with the sensor element.

While the cladding and core construction of the fiber optics allow "total internal reflection" in a closed system, various physical forces can augment the propagation of light waves. Attenuation, nonlinear phenomena, phase shift, and polarization are four characteristics that can be described as mechanisms of change in the properties of optical fibers. These characteristics determine the responsiveness of the measurands, quality of the response as well as interpretation and measurement of the data characteristics being changed throughout the fiber.

2.7.2 Fiber Optic Applications: [22]

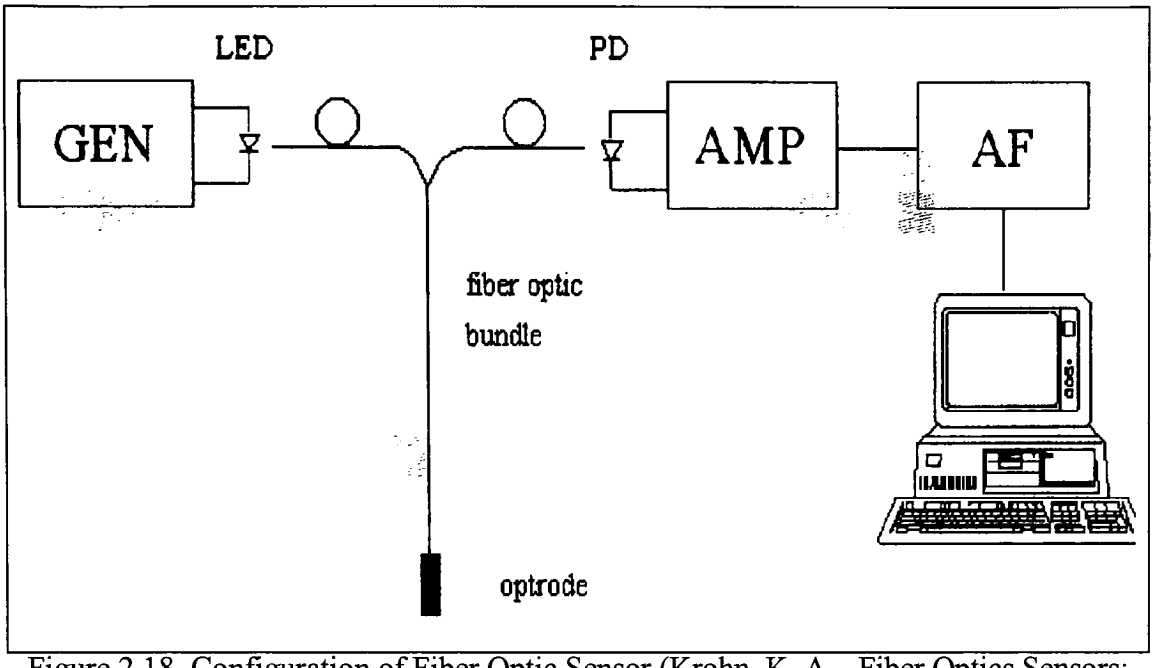


Figure 2.18, Configuration of Fiber Optic Sensor (Krohn, K. A – Fiber Optics Sensors: Fundamentals and Applications)

A generic setup for fiber optic sensors include a light source, optical wire, photo detector, processor, display optics, and traditional electronics. The various components of the Fiber Optic Chemical Sensor are connected by optical fibers. Optical fibers are thin, long cylindrical structures that allow light to propagate through the line in complete internal reflection. The light source is used to produce an optical signal through the line. The common forms of light sources in practical applications are a light emitting diode (LED). The LED is connected to an optrode with a fiber optic line. The optrode is the actual sensor that contains an indicator that predictably changes optical properties based on the substance it comes to contact with. The light signal that is emitted from the LED is usually matched to the analytical wavelength of the substance being observed to optimize the sensitivity of the sensor. The signal is then sent via fiber optic line to a photo detector. The photo detector is a piece of hardware that converts optical signals into electrical form that the data acquisition system can utilize. Once the signal is conditioned with traditional electronics, it is then send to be used by a computer.

2.7.3 Biomedical Applications: [23]

Fiber Optics Sensor has found its place in the Biomedical field. They are used in non-invasive and minimally invasive monitoring applications. Applications discussed here would be that of a Smart Scalpel and fiber optics in blood flow measurements.

"Smart Scalpel" is a dime sized, computerized laser device, which could be used in conjunction with a scalpel. The instrument contains a microscopic spectrometer and a tiny vacuum. The spectrometer analyzes the protein density of a cell and the vacuum can transport the cells through microchip-connected sensors for analysis during surgery. This process of the vacuum is the first time that fluid flow has been incorporated inside a laser micro-cavity. The fiber optics provides instant biopsies and feedback to the surgeon. A portable system incorporating the 'smart scalpel' would allow the surgeon to analyze miniscule volumes of cells and improve the success rate of surgical tumor removal. It will also decrease the costs and improve the survival rates of patients by avoiding tumor regrowth and multiple surgeries. [23]

An important field of application for non-contact measurement is that of blood velocities. The characteristics of blood flow can be used to track and monitor the health of arteries and veins. The reduction of the flow would indicate and aid in the diagnosis of circulatory disorders.

It is currently very rare to measure blood flow although it is possible to do so. The reason why the procedure is uncommon is that it is extremely invasive and involves the placement of an ultrasonic or electromagnetic sensor directly onto the point of the artery or vein that is blocked or in question. The new technique will measure the blood flow and velocity directly from inside the artery or vein. This will be accomplished by means of a small optical fiber with a diameter of a few micrometers. The sensor will have the ability to collect a series of multiple measurements and characterize the blood flow through the artery or vein. The fiber is moved through the artery or vein by means of a unique catheter that can easily be inserted and does not need to be close to the measurement point. This basket catheter maintains a stable position inside the artery by opening a

multifilament memory wire basket. Once the basket is opened the fiber has the ability to move and start taking measurements. [24]

Some major advantages of this new sensor are that it is physically very small and very light. Since this technique is extremely new there are some problems that are yet to be resolved. One is that blood cells cover the tip of the sensor and hinder its ability to collect information. To offset this effect the patient must be given anticoagulation drugs. Also the internal diameter of the artery or the vein is not easy to measure and must be estimated. This then causes uncertainty in the flow value calculated from the velocity measurements.

Acknowledgement:

I would like to acknowledge the students of EIEI 630/729 – Advanced Systems Integration (Spring 2001) whose survey projects formed the basis for the contents of this chapter.

References:

1. Detection Systems 2001 – "Photoelectric detectors", Bosch Group, 2001 <http://www.detectionsys.com/intrusion.asp?str=photo>
2. Detection Systems 2001 – "Passive Infrared Detectors", Bosch Group, 2001 < http://www.detectionsys.com/intrusion.asp?str=pir>
3. Detection Systems 2001 – "Audio Glass Break Detectors", Bosch Group, 2001 < http://www.detectionsys.com/intrusion.asp?str=glass>
4. Detection Systems 2001 – "Fire Products", Bosch Group, 2001 < http://www.detectionsys.com/fire.asp>
5. Wired or Wireless? – Advanced Detection Systems Inc., 2002 < http://www.adssecurity.com/worwless.htm>
6. Jessel, W. – "Plan and Design the Best Gas Detection", Intech Magazine, September 2000, Vol. 47 #9 <www.isa.org>
7. DGPS Explained – "Global Positioning System" – Starlink Inc., 1999 <http://www.starlinkdgps.com/dgpsexp.htm#Global%20Positioning%20System%20(GPS)>
8. DGPS Explained – "Differential Global Positioning System" – Starlink Inc., 1999 <http://www.starlinkdgps.com/dgpsexp.htm#Differential%20Global%20Positioning%20System%20(DGPS)>
9. RoyalTek GPS – "RGP – 2202", RoyalTek Company Ltd., 2000 <http://www.royaltek.com/eng/products/rgp2202.asp>
10. Il@metrohm.ch - "The Background of pH Measurement and Hints for your Daily Work", Metrohm Ltd., Version 1.0, September 1999
11. pH – "PH Theory, pH what is it?" Analytical Specialties, 2002 <http://www.analyzer.com/theory/ph/ph.htm>

12. Durafet II pH Electrodes – "Improved Durability and Reliability Increase pH Electrode Performance", Honeywell Inc., 2002 <http://www.content.honeywell.com/sensing/control/pdf/sales_lit/70-82-56-66.pdf>
13. General Information about ISFET Technology – Sentron Integrated Sensor Technology, 2002 <http://www.sentron.nl/info/isfetgen.htm>
14. Manges, W.; Allgood, G.; Smith, Stephen. – "Its Time to go Wireless", Sensors Magazine, April 1999 <http://www.sensormag.com/articles/0499/0499_10/main.shtml>
15. Wireless Technologies: Products: MiniDat – "ViaSat: Products" ViaSat, 2001 < http://www.viasat.com/products/wireless/minidat.htm>
16. Wireless Technologies: Products: MiniDat: Evolution of Wireless Data Acquisition – "ViaSat: Products" ViaSat, 2001 < http://www.viasat.com/products/wireless/minidatevol.htm>
17. Philip, B. – "Access Control by Audio-Visual Recognition" Work Study, 2000 Vol. 49 Issue 1 <http://www.emarald-library.com/brev/07949ael.htm>
18. Merriam-Webster Online – M-W Inc, 2002 < http://www.m-w.com>
19. Biometrics Working Group – "Best Practices in Testing Biometric Devices" Version 1.0
20. Desmarais, N. – "Body Language, security and E-Commerce" MCB Library Hi Tech, Vol. 18, Issue 1 <http://www.emarald-library.com/brev/2381agl.htm>
21. Murata, M. – "Handbook of Optical Fibers and Cables", 2nd Edition, New York, Marcel Dekker Inc. 1996
22. Krohn, K. A. – "Fiber Optic Sensors: Fundamentals and Applications" Instrumentation Society of America, Oct 2000 late-ed: sec. F ; pg 1 ; col. 2; Science Desk
23. Smart Scalpel to Sniff out Cancer Cells – Tech Reviews, USA Today, Associated Press, 2000 <http://www.usatoday.com/life/cyber/tech/review/crh017.htm>
24. Scalise, L.; Mul, F.; Steenbergen, W.; Anna, P. – "Recent Advances in Self-Mixing Laser-Doppler Velocitymetry: Use as a In-Vivi Blood Flow Meter" Proceedings of SPIE Vol. 3911, 2000

3. Data Acquisition Networks

Communication in automation is becoming increasingly direct, horizontally at the field level as well as vertically through all hierarchy levels in the control architecture. Depending on the application and the price industrial communication systems such as the ETHERNET/IP Network, PROFIBUS, DEVICENET, INTERBUS, and other systems offer transparent networking in all areas and levels of the automation process. Each system would be studied in detail in this chapter.

3.1 Networking Concepts: [1]

The modular networking architecture is based on two industry standard models for a layered networking architecture, namely the International Organization for Standardization (ISO) model for computer networking called the Open Systems Interconnect (OSI) Reference Model, and the Institute of Electrical and Electronic Engineers (IEEE) 802 model. The ISO-OSI and IEEE 802 models define a modular approach to networking, with each layer responsible for some discrete aspect of the networking process.

The OSI Reference Model includes seven layers:

- Application
- Presentation
- Session
- Transport
- Network
- Data-Link
- Physical

The OSI model describes the flow of data in a network, from the lowest layer (the physical connections) up to the layer containing the user's applications. Data going to and from the network is passed from layer to layer. Each layer is able to communicate with the layer immediately above it and the layer immediately below it. This way, each layer is written as an efficient, streamlined software component. When a layer receives a

packet of information, it checks the destination address, and if its own address is not there, it passes the packet to the next layer.

When two computers communicate on a network, the software at each layer on one computer assumes it is communicating with the same layer on the other computer. For example, the Transport layer of one computer communicates with the Transport layer on the other computer. The Transport layer on the first computer has no regard for how the communication actually passes through the lower layers of the first computer, then across the physical media, and finally through the lower layers of the second computer.

· The ***Application layer*** represents the level at which applications access network services. This layer represents the services that directly support applications such as software for file transfers, database access, and electronic mail.

· The ***Presentation layer*** translates data from the Application layer into an intermediary format. This layer also manages security issues by providing services such as data encryption, and compresses data so that fewer bits need to be transferred onto the network.

· The ***Session layer*** allows two applications on different computers to establish, use, and end a session. This layer establishes dialog control between the two computers in a session, regulating which side transmits, and when and how long it transmits.

· The ***Transport layer*** handles error recognition and recovery. It also repackages long messages when necessary into small packets for transmission and, at the receiving end, rebuilds packets into the original message. The receiving Transport layer also sends receipt acknowledgments.

· The ***Network layer*** addresses messages and translates logical addresses and names into physical addresses. It also determines the route from the source to the destination

computer and manages traffic problems, such as switching, routing, and controlling the congestion of data packets.

· The ***Data Link layer*** packages raw bits from the Physical layer into frames (logical, structured packets for data). This layer is responsible for transferring frames from one computer to another without errors. After sending a frame, it waits for an acknowledgment from the receiving computer.

· The ***Physical layer*** transmits bits from one computer to another and regulates the transmission of a stream of bits over a physical medium. This layer defines how the cable is attached to the network adapter and what transmission technique is used to send data over the cable.

The ISO OSI is the basis of communication in any form of network. The various industrial communication networks are developed around these fundamentals.

3.2 ETHERNET/IP NETWORK: [2]

Ethernet Industrial Protocol (EtherNet/IP) is an open industrial networking standard that supports implicit messaging (real-time I/O messaging), explicit messaging (message exchange) or both and uses commercial off-the-shelf Ethernet communication chips and physical media. Because Ethernet technology has been used since the mid 1970s and is widely accepted throughout the world, Ethernet products serve a large community of vendors. By using Ethernet products, not only do we follow the common trend in technology today but we also have the ability to access device-level data all the way from the Internet. EtherNet/IP emerged due to the high demand for using the Ethernet network for control applications.

EtherNet/IP is an open network because it uses:

- IEEE 802.3 Physical and Data Link standard.
- Ethernet TCP/IP protocol suite (Transmission Control Protocol/Internet Protocol), the Ethernet industry standard.

- Control and Information Protocol (CIP) the protocol that provides real-time I/O messaging and information / peer-to-peer messaging.

TCP/IP is the transport and network layer protocol of the Internet and is commonly linked with Ethernet installations and the business world. TCP/IP provides a set of services that any two devices can use to share data. Because Ethernet technology and standard protocol suites such as TCP/IP have been published for public use, standardized software tools and physical media have been mass-produced and are readily available offering two benefits:

- Known technology
- Accessibility

The UDP/IP (User Datagram Protocol) is also used in conjunction with the Ethernet network. UDP/IP provides fast, efficient data transport necessary for real-time data exchange.

To make EtherNet/IP successful, CIP has been added on top of TCP/UDP/IP to provide a common application layer. Therefore, when one chooses an EtherNet/IP product, one is choosing a product with CIP capabilities. Additionally, EtherNet/IP uses the producer/consumer network model. With the introduction of Ethernet switch technology and full-duplex data transmission, the occurrence of data collisions is theoretically eliminated, and performance is drastically enhanced on an EtherNet/IP network.

Typical devices communicating across an EtherNet/IP network include:

- Mainframe/Mini/ Micro Computers
- PLC processors
- Robots
- HMI
- I/O and I/O adapters

Target applications include:

- Plant management system tie-in, material handling
- Configuration, data collection, and control on a single high-speed network
- Time-critical applications with no established schedule

3.2.1 Mechanical Design:

Typically, an EtherNet/IP network uses an active star topology in which groups of devices are connected point-to-point to a switch as shown in Figure 3.1. The benefit of a star topology is in its support of both 10 and 100M bit/s products. One could mix 10 and 100M bit/s products, and most Ethernet switches will negotiate the speed. The star topology offers connections that are simple to wire, easy to debug and detect failures, and easy to maintain.

EtherNet/IP is designed to handle large amounts of messaging data, 1500 bytes maximum per packet. In addition to handling large amounts of data, the EtherNet/IP speed (10/100M bit/s) makes that data transmission even more appealing. Because of the wide acceptance of Ethernet technology throughout the years, the cost per node for Ethernet switches and other Ethernet physical media is rapidly decreasing. With these characteristics, EtherNet/IP is becoming a viable choice for many control applications.

The EtherNet/IP cabling components provide flexibility in two areas: overall cost and manufacturer. Due to the large number of third-party vendors, one now will have a wide selection of media components and cost considerations.

Standard twisted-pair and fiber-optic cable are fully functional with EtherNet/IP. Depending on the network configuration, one may need an Ethernet hub or switch. A **hub** is an inexpensive connectivity method that provides an easy method of connecting devices on information networks. A **switch** reduces collisions and is recommended for real-time control installations. **Routers** are used to isolate control data traffic from other types of office data traffic, and to isolate information traffic on the plant floor from process control traffic. Routers are also used for security purposes, i.e., firewalls.

Repeaters extend overall network cable length. They can also connect networks with different media types.

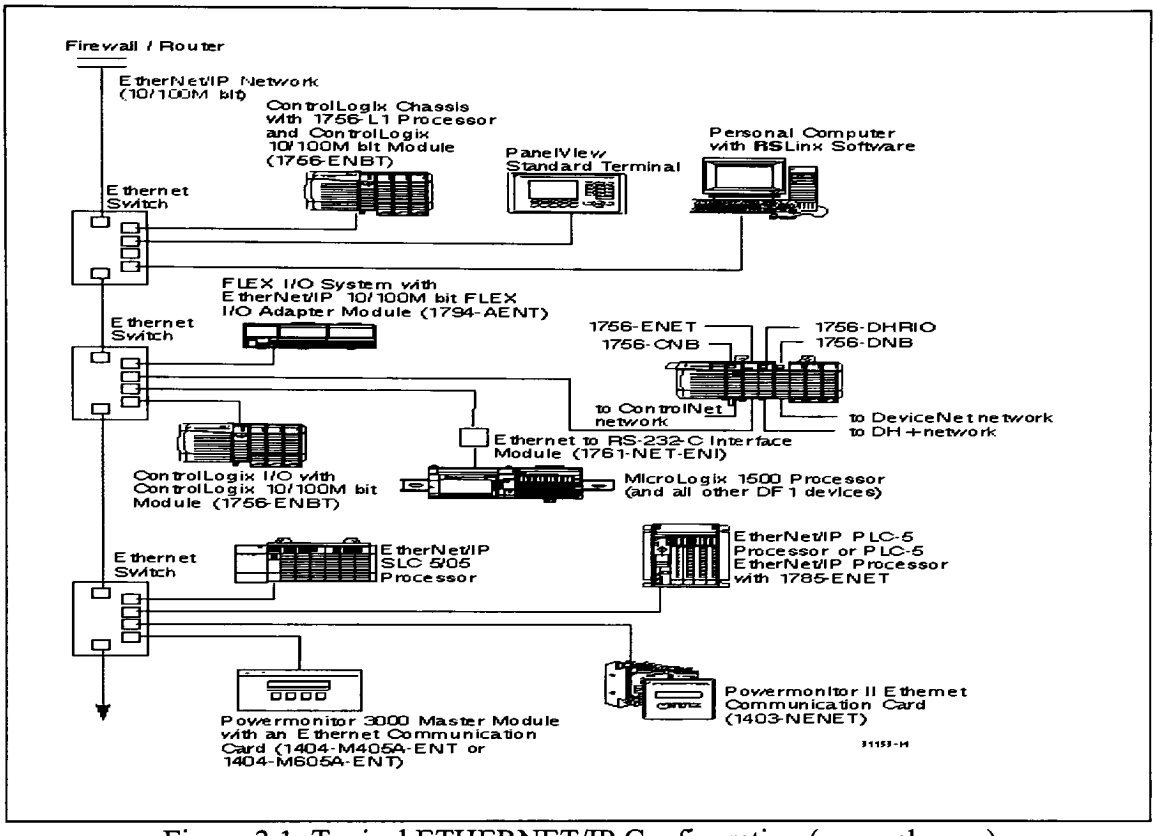


Figure 3.1, Typical ETHERNET/IP Configuration (www.ab.com)

3.2.2 EtherNet/IP Transmission Types:

The control portion of CIP is used for real-time I/O messaging or implicit messaging. The information portion of CIP is used for message exchange or explicit messaging. The three definitions given below will help explain the transmission types used in the table.

- *Information:* Non-time critical data transfers typically large packet size. Information data exchanges are short-lived explicit connections between one originator and one target device. Information data packets use the TCP/IP protocol and take advantage of the TCP data handling features.
- *I/O Data:* Time-critical data transfers - typically smaller packet size. I/O data exchanges are long-term implicit connections between one originator

and any number of target devices. I/O data packets use the UDP/IP protocols and take advantage of high-speed throughput capability of UDP.

Real-time Interlocking: Cyclic data synchronization between one producer processor and any number of consumer processors. Interlocking data packets use the faster UDP/IP protocols and take advantage of high-speed throughput capability of UDP.

EtherNet/IP Transmission Types	Message Type	Description	Example
Information	Explicit	Non-time-critical Information Data	Read / Write data via message instruction
I/O Data	Implicit	Real-time I/O Data	Control real-time data from a remote I/O device
Real-time Interlocking	Implicit	Real-time Device Interlocking	Exchange real-time data between two processors

Table 3.1, EtherNet/IP Transmission Types

3.2.3 Features of EtherNet/IP:

- Worldwide acceptance of Ethernet products
- Near elimination of data collisions through the use of switch technology and full-duplex transmission
- Support of both 10 and 100M bit/s products
- Products with built-in web server capability

3.3 PROFIBUS: [3]

PROFIBUS is a vendor-independent, open field bus standard for a wide range of applications in manufacturing and process automation. Vendor-independence and openness are ensured by the international standards EN 50170, EN 50254 and IEC 61158.

PROFIBUS allows communication between devices of different manufacturers without any special interface adjustment. PROFIBUS can be used for both high-speed time critical applications and complex communication tasks.

PROFIBUS offers functionally graduated communication protocols: **DP and FMS**. Depending on the application, the transmission technologies RS-485, IEC 1158-2 or fiber optics are available. In the course of further technical development, the PROFIBUS User Organization is currently working on the implementation of universal concepts for vertical integration on the basis of Ethernet TCP/IP.

3.3.1 Communication Basis:

PROFIBUS defines the technical characteristics of a serial field bus system with which distributed digital programmable controllers can be networked from field level to cell level. PROFIBUS is a multi-master system and thus allows the joint operation of several automation, engineering or visualization systems with their distributed peripherals on one bus. PROFIBUS distinguishes between the following types of devices:

- **Master devices** determine the data communication on the bus. A master can send messages without an external request when it holds the bus access rights (the token). Masters are also called active stations.
- **Slave devices** are peripherals such as I/O devices, valves, drives and measuring transducers. They do not have bus access rights and they can only acknowledge received messages or send messages to the master when requested to do so. Slaves are called passive stations. Since they only require a small portion of the bus protocol, their implementation is particularly economical.

The DP Communication Profile is designed for efficient data exchange at the field level. The central automation device, such as a PLC/PC or process control systems, communicates through a fast serial connection with distributed field devices such as I/O, drives and valves, as well as measuring transducers. Data exchange with the distributed devices is mainly cyclic. The communication functions required for this are defined by the basic DP functions in accordance with EN 50170. In addition to these basic functions, DP also offers extended acyclic communication services for the parameterization, operation, monitoring and alarm handling of intelligent field devices.

The extended DP functions make it possible to transmit acyclic read and write functions as well as alarms between master and slaves, parallel and independent of cyclic user data communication. This allows the user to use, for example, an engineering tool (DPM2) to optimize the device parameters of the connected field devices (slaves) or reads out the device status without disturbing system operation.

With these extended functions, DP meets the requirements of even complex devices, which often have to be parameterized during operation. Nowadays, the extended DP functions are mainly used for the online operation of the PA field devices by means of engineering tools. Transmission of the acyclic required data is performed with a lower priority parallel to the high-speed cyclic user data transfer. The master requires some additional time to carry out the acyclic communication services. This must be taken into account in the parameterization of the overall system. To achieve this, the parameterization tool usually increases the token circulation time somewhat in order to give the master a chance to carry out not only cyclic data transmission but also acyclic communication tasks.

3.3.2 PROFIBUS Technology in Automation:

Communication systems such as the Ethernet-based PROFINET, the fieldbus PROFIBUS and other systems like the sensor/actuator bus AS-Interface offer the ideal preconditions for transparent networking in all areas and levels of the automation process.

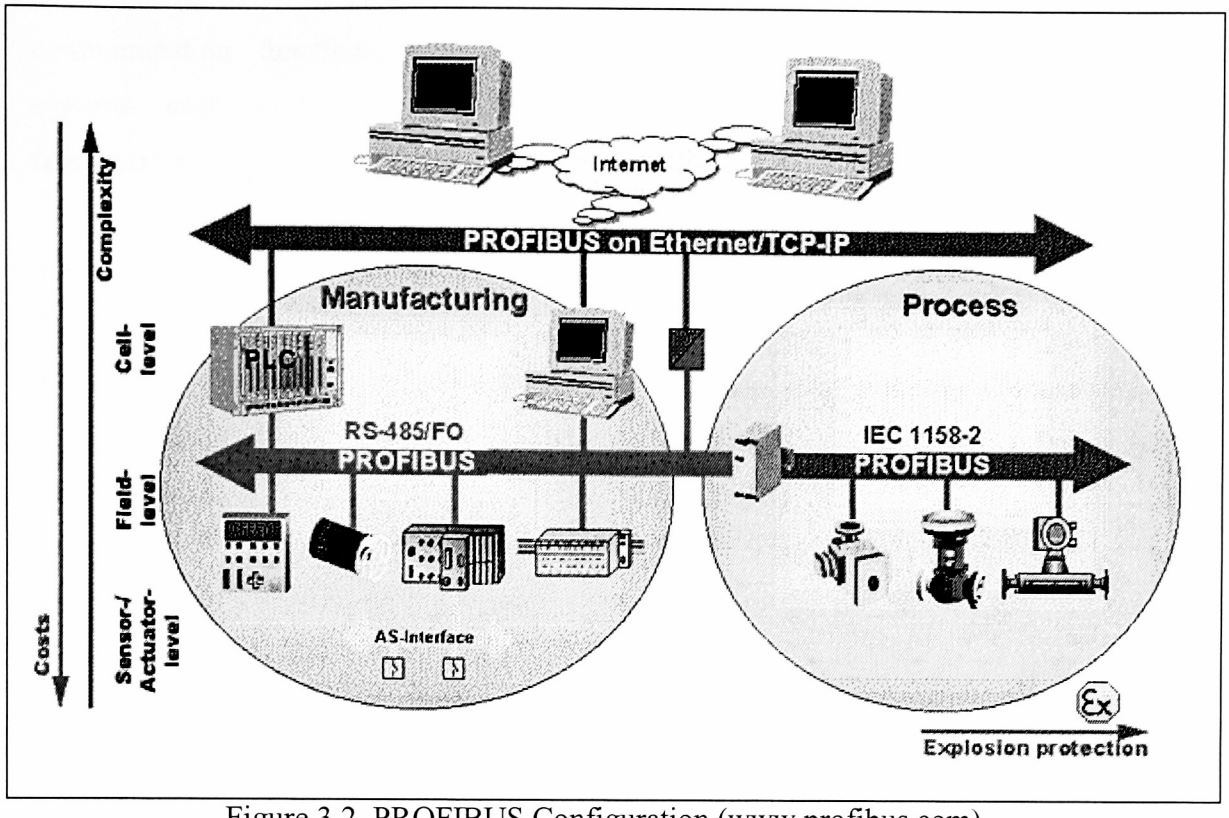


Figure 3.2, PROFIBUS Configuration (www.profibus.com)

At the **sensor/actuator level,** signals from binary sensors and actuators are transmitted via a sensor/actuator bus. It is a particularly simple, low-cost installation technique, through which data and a 24-volt power supply for the end devices are transmitted using a common medium. The data are transmitted purely cyclically. AS-Interface is a suitable bus system for this field of applications.

At **field level** the distributed peripherals such as I/O modules, measuring transducers, drive units, valves, and operator terminals communicate with automation systems via an efficient, real-time communication system. The transmission of the process data is effected cyclically, while alarms, parameters and diagnostic data also have to be transmitted acyclically if necessary. PROFIBUS meets these requirements and offers a transparent solution for manufacturing as well as for process automation.

At **cell level**, the programmable controllers such as PLC and IPC communicate with each other. The information flow requires large data packets and a large number of powerful

communication functions. Smooth integration into company-wide communication systems, such as Intranet and Internet via TCP/IP and Ethernet are important requirements.

3.3.3 FMS Communication Profile:

The FMS Communication Profile is designed for communication at cell level. At this level, programmable controllers (PLCs and PCs) communicate primarily with each other. In this application area a high degree of functionality is more important than fast system reaction times.

The FMS application layer (7) consists of the following parts:

- The Fieldbus Message Specification (FMS) and
- The Lower Layer Interface (LLI)

The PROFIBUS-FMS communication model permits distributed application processes to be unified into a common process by using communication relationships. That portion of an application process in a field device, which can be reached via communication, is called a Virtual Field Device (VFD). Figure 3.3 shows the relationship between the real field device and the virtual field device. In this example, only certain variables (i.e., number of units, rate of failure and downtime) are part of the virtual field device and can be read or written via the two communication relationships. The variables Required Value and Recipe are not available with FMS.

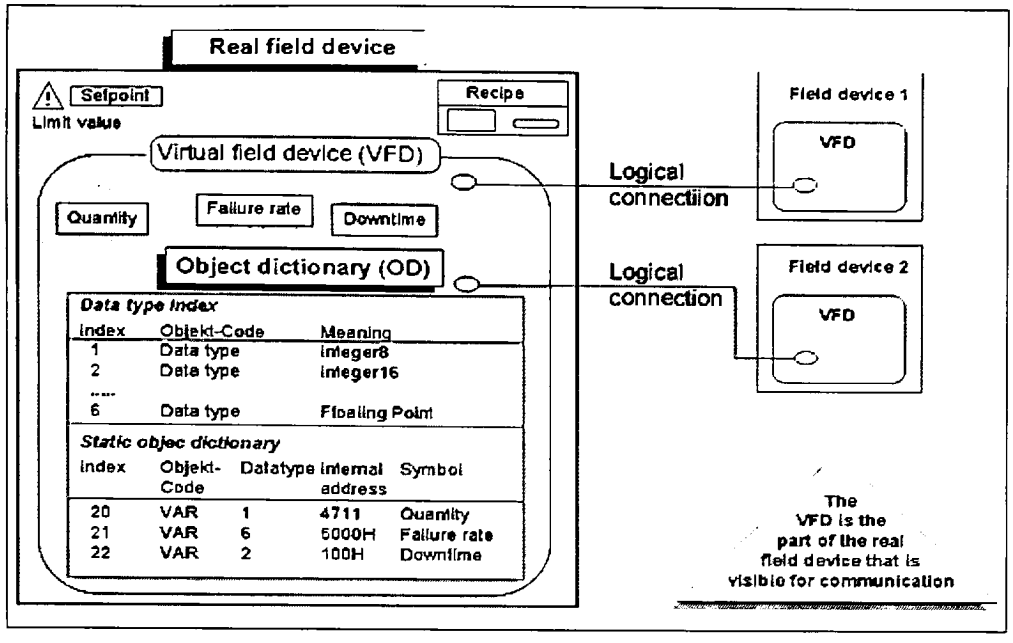


Figure 3.3, Virtual Field Device (VFD) with Object Dictionary (OD)
(www.profibus.com)

All **communication objects** of an FMS device are entered in the Object Dictionary (OD). The object dictionary contains description, structure and data type, as well as the relationship between the internal device addresses of the communication objects and their designation on the bus (index/name).

Static communication objects are entered in the static object dictionary. They are configured once and cannot be modified during operation. FMS recognizes five types of communication objects:

- Simple Variable
- Array (series of simple variables of the same type)
- Record (series of simple variables of different types)
- Domain
- Event (event message)

Dynamic communication objects are entered in the dynamic section of the object dictionary, which can be modified during an operation.

Logical addressing is the preferred method of addressing for the objects. Accessing is performed with a short address (the index), which is a number of type Unsigned16. Each object has a unique index. An additional option is to address the objects by name. Communication objects can also be protected from unauthorized access through **access protection**, or the permitted services for accessing an object (e.g., read only) can be restricted.

3.4 DEVICENET: [4, 5]

DeviceNet is a low-cost industrial network to connect industrial devices such as limit switches, photoelectric cells, valve manifolds, motor starters, drives, and operator displays to PLCs and PCs. The network eliminates expensive hard wiring while providing device-level diagnostics. DeviceNet offers robust, efficient data handling because it is based on the Producer/Consumer technology. This modern communications model offers key capabilities that allow the user to effectively determine what information is needed and when.

DeviceNet communication link is based on a broadcast-oriented, communications protocol – the Controller Area Network (CAN). The CAN protocol was originally developed by BOSCH for the European automotive market for replacing expensive, wire harnesses with low-cost network cable on automobiles. As a result, the CAN protocol has fast response and high reliability for applications as demanding as control of anti-lock brakes and air bags. Chips are available in a variety of packages with high temperature ratings and high noise immunity, attributes that are well suited for the industrial automation market as well

3.4.1 DeviceNet Specification:

The DeviceNet specification defines a network communication system for moving data between elements of an industrial control system. The specification is divided into two volumes and defines the following elements:

Volume 1:

- DeviceNet Communication Protocol and Application (Layer 7 – Application)

- CAN and its use in DeviceNet (Layer 2 – Data Link Layer)
- DeviceNet Physical Layer and Media (Layer 1 – Physical Layer)

Volume 2:

- Device Profiles to obtain interoperability and interchangeability among like products

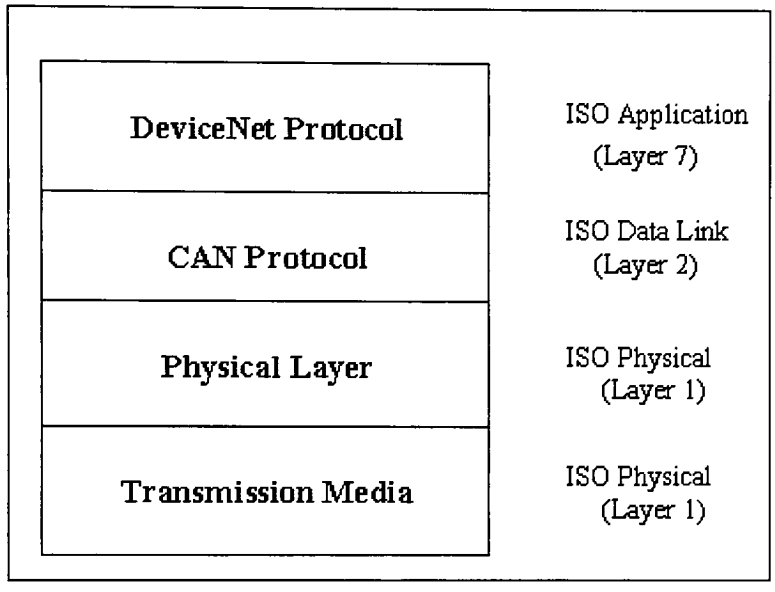


Figure 3.4, DeviceNet is an Application Level Protocol (www.odva.org)

CAN Communication Protocol Features are:

- Peer-to-Peer data exchange in which any DeviceNet product can produce and consume messages
- Master/Slave operation defined as a proper subset of Peer-to-Peer
- A DeviceNet product may behave as a Client or a Server or both
- A DeviceNet network may have up to 64 Media Access Control Identifiers pr MAC Ids (node addresses). Each node can support an infinite number of I/O. Typical I/O counts for pneumatic valve actuators are 16 or32.

3.4.2 DeviceNet Physical Layer and Media:

The variety of topologies that are possible is shown in Figure 3.5. The specification also deals with system grounding, mixing thick and thin media, termination, and power distribution.

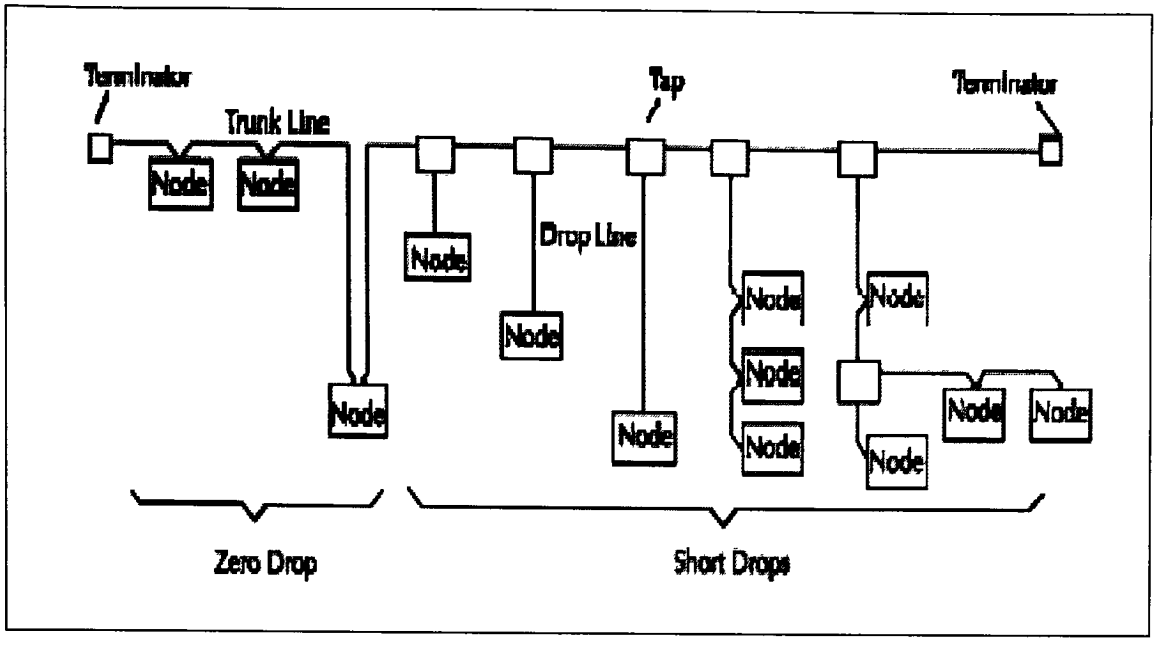


Figure 3.5, DeviceNet Physical Layer and Media (www.odva.org)

The basic trunkline – dropline topology provides separate twisted pair busses for both signal and power distribution. Thick or thin cable can be used for either trunklines or droplines. End-to-end network distance varies with data rate and cable size.

Data Rates	125 Kbps	250 Kbps	500 Kbps
Thick Trunk Length	500m	250m	100m
Thin Trunk Length	100m	100m	100m
Maximum Drop Length	6m	6m	6m
Cumulative Drop Length	156m	78m	39m

Table 3.2, End-to-end network distance varies with data rate and cable thickness

Devices can be powered directly from the bus and communicate with each other using the same cable. Nodes can be removed or inserted from the network without powering down the network.

Power taps can be added at any point in the network, which makes redundant power supplies possible. The trunkline rating is 8amps. An opto-isolated design option allows externally powered devices (e.g., AC Drives starters and solenoid valves) to share the

same bus cable. Other CAN-based networks allow only a single power supply for the entire network.

3.4.3 DeviceNet Communication Profile: [6]

The Data Link Layer of DeviceNet is completely defined by the CAN specification and by the implementation of CAN Controller chips. The CAN specification defines two bus states called dominant (logic 0) and recessive (logic 1). Any transmitter can drive the bus to a dominant state. The bus can only be in the recessive state when no transmitters in the dominant state.

Several frame types are defined by CAN:

- Data frame
- Overload frame
- Remote frame
- Error frame

Data are moved on DeviceNet using data frame. The other frames are either not used on DeviceNet are for exception handling. Higher priority data gets the right of way – DeviceNet is similar to Ethernet, where any DeviceNet node can attempt to transmit if the bus is quiet. This provides inherent peer-to-peer capability. If two or more nodes try to access the network simultaneously, a bit-wise non-destructive arbitration mechanism solves the potential conflict with no loss of data or bandwidth. By comparison, Ethernet uses collision detectors, which result in loss of data or bandwidth, as both nodes have to back off and retransmit their data.

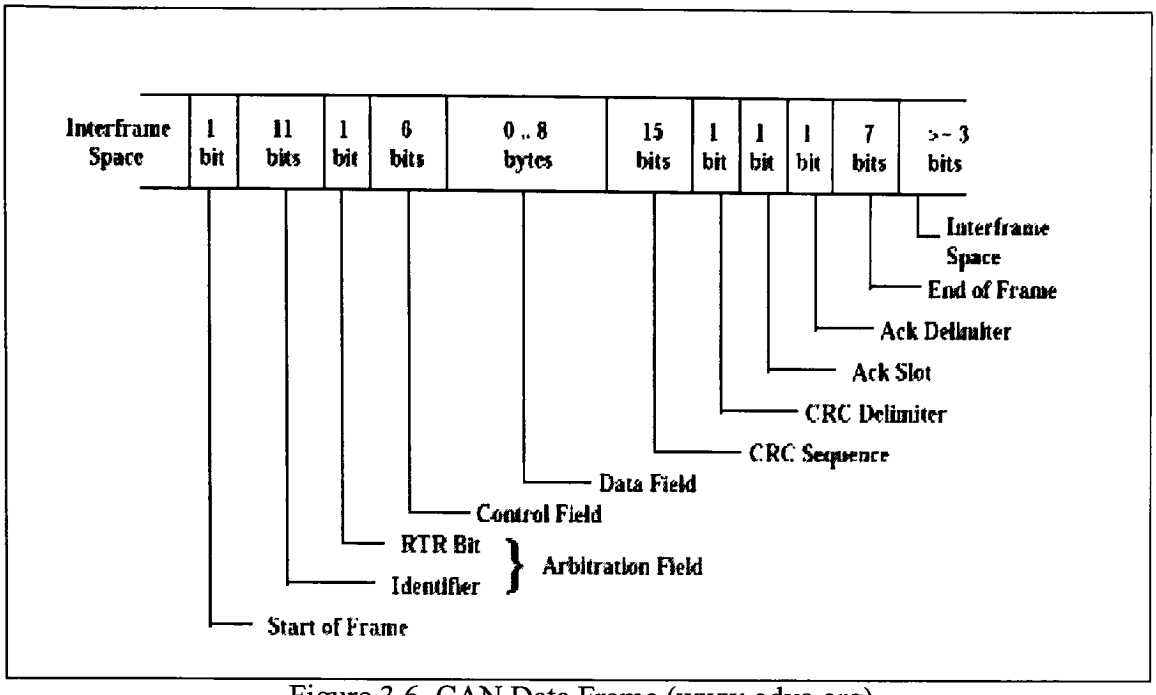


Figure 3.6, CAN Data Frame (www.odva.org)

CAN use a bit-wise arbitration method of collision resolution. All receivers on a CAN network synchronize to the transition from recessive to dominant represented by a Start of Frame bit. The identifier and the RTR (Remote Transition Request) bit together form the Arbitration Field. The Arbitration Field is used to facilitate media access. Since DeviceNet does not use the RTR bit for any purpose it does not enter into bus access priority consideration. When a device transmits, it also monitors (receives) what it sends to make sure that it is the same. If a node transmitting a recessive bit receives a dominant bit while sending the arbitration field, it stops transmitting. The winner of an arbitration between two nodes transmitting simultaneously is the one with the lower numbered 11-bit identifier.

The CRC field is a cyclic redundancy check word, which used by CAN controllers to detect frame errors. It is computed from the bits that come before it. A dominant bit in the ACK slot means at least one receiver besides the transmitter heard the transmission.

3.4.4 DeviceNet Communication Protocol and Application:

Applications using DeviceNet combine standard or application specific objects together into what is called a Device Profile. The Device Profile fully defines the device as viewed from the network.

DeviceNet supports strobed, polled, cyclic, change-of-state, and application-triggered data movement. The user can choose master/slave, multi-master and peer-to-peer or a combination configuration depending on device capability and application requirements. The choice of data movement can significantly speed up system response time.

The DeviceNet Communication Protocol is based on the idea of connections. One must establish a connection in order to exchange information with that device. To establish a connection each product will implement either an Unconnected Message Manager (UCMM) or an Unconnected Port. Both perform their function by reserving some of the available CAN identifiers.

When either the UCMM or the Unconnected Port is used to establish an Explicit Messaging Connection, that connection is then used to move information from one node to another, or to establish additional I/O connections. Once connections have been established, I/O data may be moved among devices on the network. At this point, all the protocol of the DeviceNet I/O message is contained within the 11- bit CAN identifier, while everything else in the data frame is data.

3.5 INTERBUS: [7, 8]

INTERBUS is the fieldbus from PHOENIX CONTACT. Development started in 1985 to replace the complex parallel individual cabling of devices in the field with a serial fieldbus. Simple sensors and actuators as well as intelligent sensor and actuator systems can be connected to INTERBUS.

INTERBUS is a ring system with a maximum of 512 stations. The ring system offers the following advantages:

- No terminal resistors, the ring is closed automatically
- No address setting of the devices on site
- Easy to extend, the ring is extended "automatically".

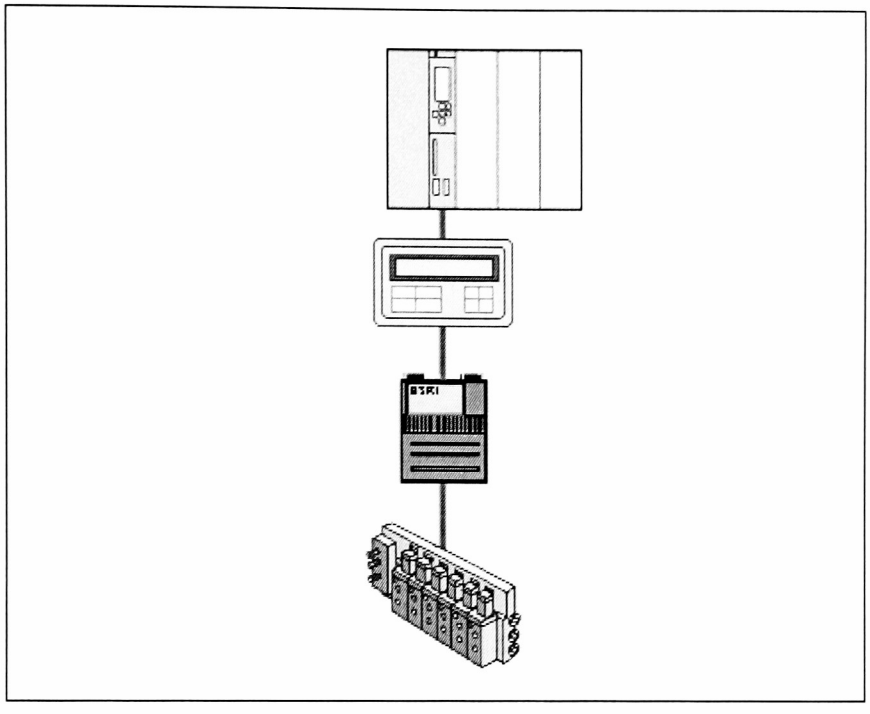


Figure 3.7, Ring Topology (www.interbusclub.com)

The INTERBUS master/slave system enables the connection of up to 512 devices across 16 levels of the network, where the last device in the network automatically closes the ring. This point-to-point connection eliminates the need for termination resistors. The system can adapt flexibly to meet the users requirements by adding or removing devices.

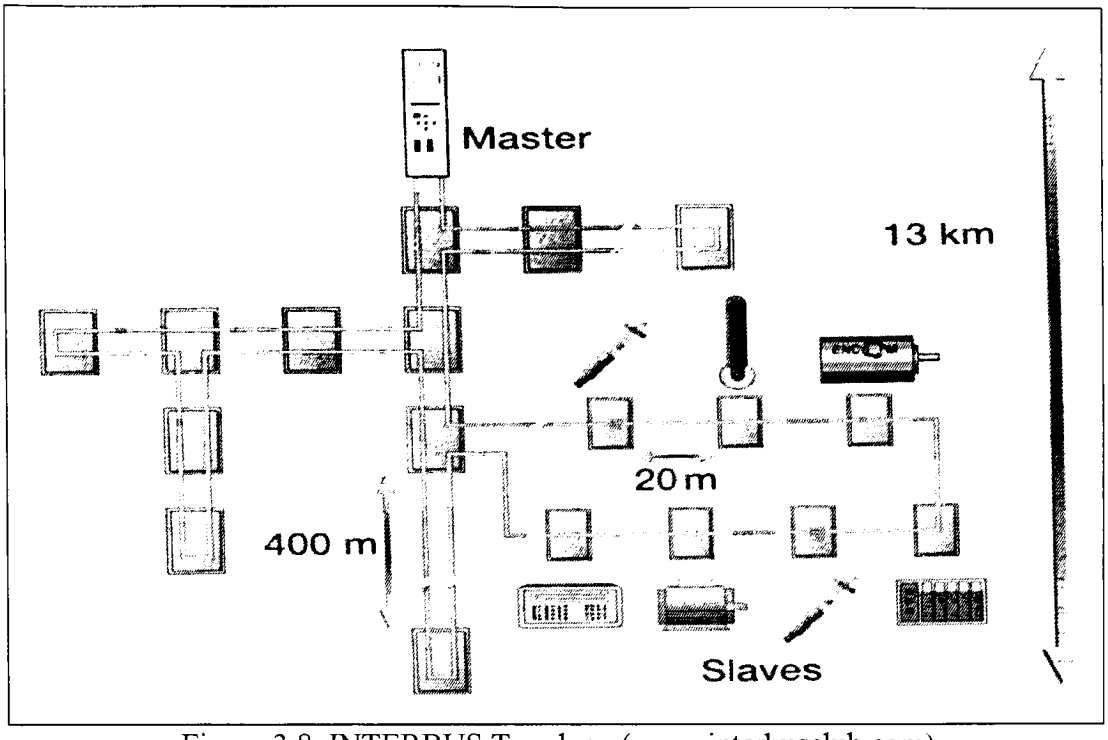


Figure 3.8, INTERBUS Topology (www.interbusclub.com)

The disadvantage of this type of connection scheme is that one failed connection would disable the entire network and the limited ability to transfer large amounts of data.

3.5.1 Physical Addressing:

Unlike other systems where data is assigned by entering a bus address using DIP or rotary switches on individual device, in the INTERBUS system data is automatically assigned to devices using their physical location in the system. This plug and play feature is a great advantage with regard to the installation effort and service-friendliness of the system. The problems and errors, which may occur when manually setting device addresses during installation and servicing, are often underestimated. The ability to assign "easy to understand" software name to the physical addresses, allows devices to be added or removed without re-addressing existing devices.

3.5.2 Basic Elements of INTERBUS:

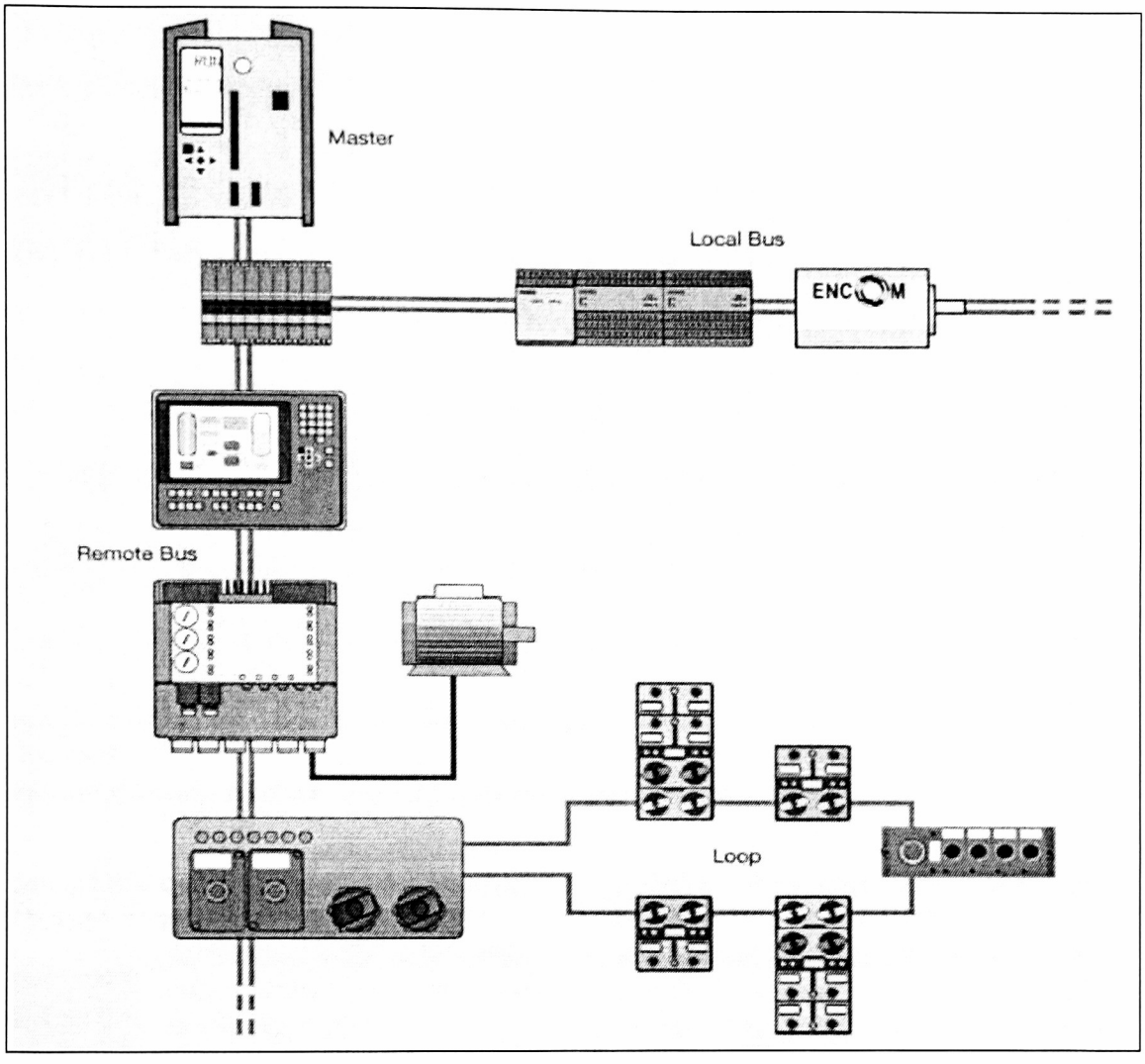


Figure 3.9, Individual Components on an INTERBUS Network (www.interbusclub.com)

3.5.2.1 Controller Board:

The controller board is the master that controls the data traffic. It transfers output data to the corresponding modules, receives input data, and monitors data transfer. In addition, diagnostic message are displayed and error messages are transmitted to the host system.

3.5.2.2 Remote Bus:

The controller board is connected to the remote devices via the remote bus. A branch from this connection is referred to as a remote bus branch. Data can be physically transmitted via copper cables (RS-485 standard), optical fibers, infrared transmission

paths, slip rings or other media. Special bus terminal modules and certain I/O modules or devices such as robots, drives or operating devices can be used as remote bus devices. Each has a local voltage supply and an electrically isolated outgoing segment.

3.5.2.3 Bus Terminal:

The bus terminal modules, or devices with embedded bus terminal functionality, are connected to the remote bus. The distributed local buses branch out of the bus terminal module with I/O modules, which establish the connection between INTERBUS and the sensors and actuators. The bus terminal divides the system into individual segments, allowing to switch branches, ON/OFF separately during operation. The module electronics for the connected I/O modules can be supplied with power from this source. The bus terminal amplifies the data signal (repeater function) and electrically isolates the bus segments.

3.5.2.4 Local Bus:

The local bus branches from the remote bus via a bus interface module and connects the local bus devices. Branches are not allowed at this level. The bus terminal module supplies the communications power, while the switching voltage for the outputs is applied separately at the output modules. Local bus devices are typically I/O modules in a distributed substation structure.

3.5.2.5 Loop:

Distributed sensors and actuators on machines or systems are networked with the INETRBUS loop. The two-wire, unshielded cable simultaneously transports data and supplies power to the connected devices. There are also various INTERBUS modules, which are tailored to specific tasks, such as motor starters.

3.5.3 Data Transmission:

INTERBUS is the only bus system working according to the summation frame method that uses only one protocol frame for messages from all the devices.

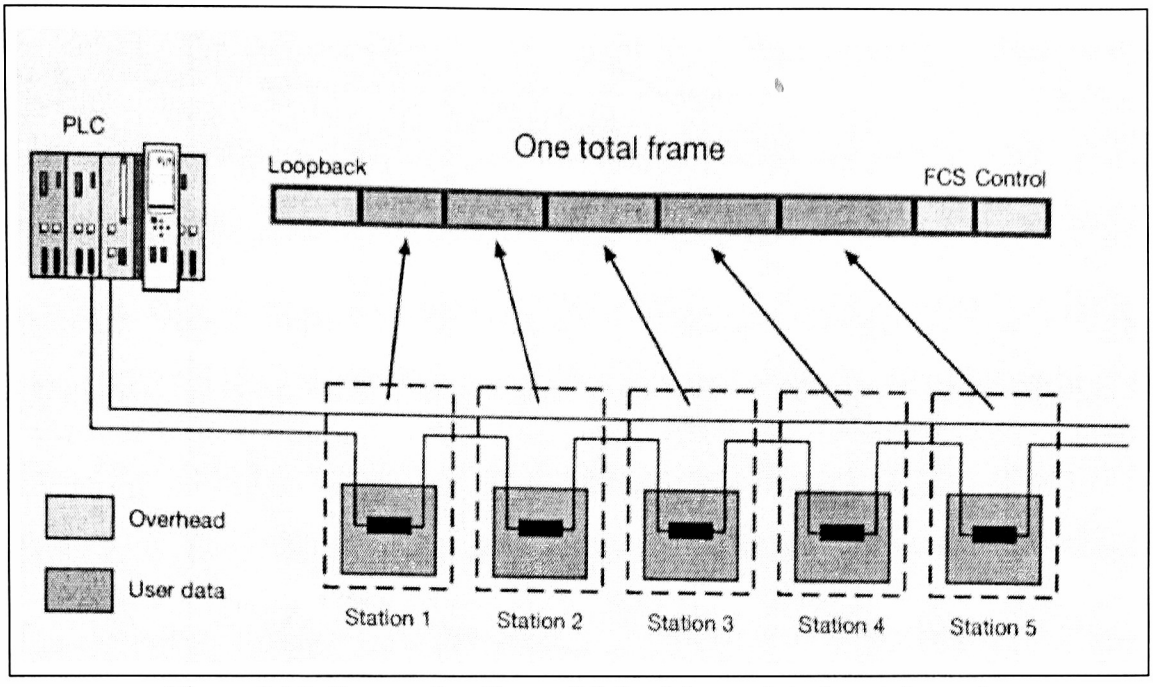


Figure 3.10, Summation Frame Method (www.interbusclub.com)

In this master/slave access method, the bus master acts as the coupling to the higher-level control or bus system.

The summation frame method provides a high level of efficiency during data transmission and enables data to be sent and received simultaneously (full duplex operation). With this data transmission method, INTERBUS ensures constant and predictable sampling intervals for setpoints and real time control values.

A summation frame consists of the header, the loop-back word, and data save and end information; data from all the connected I/O devices is grouped together in a block. The additional information that is required is transmitted only once per cycle. In practice this method can be considered as a register, which is formed by the devices that are connected in a ring system. In INTERBUS, register consists of a number of binary memory cells, which push digital information from cell to cell at clock pulses. Each device has a certain number of buffers assigned to a preset number of cells for different tasks, e.g., data input and output for the process. Additional registers monitor the data transmission for errors. An INTERBUS device contains three registers that are connected in parallel. I/O data is

transferred using the data register. The type of INTERBUS device is defined in the identification register. This enables the bus master to identify the devices and the bus topology, as well as to carry out addressing. Data is saved using the CRC16 register (cyclic redundancy check), where the received data is checked for correctness.

The cycle time i.e., the time required for I/O data to be exchanged once with all the connected modules, depends on the amount of user data in the INTERBUS system. The cycle time increases linearly with the number of I/O points, because it depends in the amount of data to be transmitted. A certain amount of time is needed for each bit. Because the summation frame has a set length, the cycle time also remains constant. In INTERBUS the deterministic method of operation is provided by the summation frame method, which is essential for fast controllers.

Process data that is to be sent to the I/O devices is stored in the output buffer of the master in the physical order of the connected output stations. During data output, process information in the form of input data is simultaneously returned to the input buffer of the master. Once the entire summation frame has been sent and simultaneously read in again, all output data is correctly positioned in the individual devices.

A network is established by connecting all the devices, whose length and structure corresponds to the structure of the user data field in the summation frame telegram. The amount of user data for the summation frame method is over 60%. Bus access conflicts do not occur due to the master/slave structure. This means that potential error sources are avoided in the outset.

3.6 Network Comparison Charts: [9]

3.6.1 Physical Characteristics:

Fieldbus Name	Network Topology	Physical Media	Max. Devices (nodes)	Max. Distance
PROFIBUS DP/PA	Line, star & ring	Twisted-pair or fiber	127 nodes (124 slaves - 4 seg, 3 rptrs) + 3 masters	100m between segments @ 12Mbaud; 24 Km (fiber) (baudrate and media dependent)
INTERBUS	Segmented with "T" drops	Twisted-pair, fiber, and slip-ring	256 nodes	400 m/segment, 12.8 Km total
DeviceNet	Trunkline/dropline with branching	Twisted-pair for signal & power	64 nodes	500m (baudrate dependent) 6Km w/ repeaters
Industrial Ethernet	Bus, Star, Daisy-Chain	Thin Coax, Twisted Pair, Fiber; Thick Coax (rare)	1024 nodes, expandable to more via Routers	Thin: 185m 10 Base T (Twisted Pair): Max 100m long (90 meters horizontal cable, 5m drops, 1m patch)

3.6.2 Transport Mechanism:

Fieldbus Name	Communication Methods	Transmission Properties	Data Transfer Size	Arbitration Method	Error Checking
PROFIBUS DP/PA	Master/slave peer to peer	DP: 9.6, 19.2, 93.75, 187.5, 500 Kbps, 1.5, 3, 6, 12 Mbps PA: 31.25 kbps	0-244 bytes	Token passing	HD4 CRC
INTERBUS	Master/slave with total frame transfer	500kBits/s, full duplex	1-64 Bytes data 246 Bytes Parameter 512 bytes h.s. unlimited block	None	16-bit CRC
DeviceNet	Master/slave, multi-master, peer to peer	500 kbps, 250 kbps, 125 kbps	8-byte variable message with fragmentation for larger packets	Carrier-Sense Multiple Access w/ Non-Destructive Bitwise Arbitration	CRC check
Industrial Ethernet	Peer to Peer	10, 100Mbps	46-1500 Bytes	CSMA/CD	CRC 32

3.6.3 Performance:

Fieldbus Name	Cycle Time: 256 Discrete 16 nodes with 16 I/Os	Cycle Time: 128 Analog 16 nodes with 8 I/Os	Block transfer of 128 bytes 1 node
PROFIBUS DP/PA	Configuration dependent typically <2ms	Configuration dependent typically <2ms	Not available
INTERBUS	1.8 ms	7.4 ms	140 ms
DeviceNet	2.0 ms Master-slave polling	10 ms Master-slave polling	4.2 ms
Industrial Ethernet	Application Layer Dependent	Application Layer Dependent	Application Layer Dependent

References:

1. James Bond Meets the 7 Layer OSI Model - www.pe.net, May 1997 <http://www.pe.net/~rlewis/Resources/james.html>
2. EtherNet/IP Network Rockwell Automation, 2002 <http://www.ab.com/catalogs/b113/comm/ethernet.html>
3. Technology Description Overview – Profibus International, 2001 <http://www.profibus.com/technology/docu.html>
4. ODVA – The Open DeviceNet's Vendor Association, 2002 <http://www.odva.org/10_2/00_index.htm>
5. Welcome to DeviceNet/ODVA Official Website - DEVICENET Technical Overview, 2002 <http://www.odva.org/10_2/05_fp_tech.htm>
6. Romito, R – "DeviceNet Technical Overview (Presentation)", 2002 <http://www.odva.org/10_2/09_down/ray_romito_files/frame.htm>
7. INTERBUS Online – INTERBUS, 2002 <http://www.interbusclub.com>
8. INTERBUS Technical Overview – INTERBUS, 2002 < http://www.interbusclub.com/en/doku/pdf/interbus_basics_de.pdf>
9. The Synergetic Fieldbus Comparison Chart – Synergetic Micro Systems, Inc, 1999 <http://www.synergetic.com/compare.htm>

4. Introduction to LabVIEW

4.1 Introduction:

LabVIEW is short for Laboratory Virtual Instrument Engineering Workbench. It is a development application developed and introduced by National Instruments, who made it a powerful and flexible instrumentation and analysis software.[1] It is highly productive software, which uses the graphical programming environment that combines easy to use graphical development with the flexibility of a powerful programming language.[2] LabVIEW provides a visual programming setup that allows for straightforward implementation of an algorithm or a control equation, making it simpler for people to execute a programming task even though they are not experts in computer programming.[1]

Engineers and scientists in research, development, production, test, and service industries as diverse as automotive, semiconductor, aerospace, electronics, chemical, telecommunications, and pharmaceutical have used and continue to use LabVIEW to support their work. LabVIEW is a major player in the area of testing and measurements, industrial automation, and data analysis.[1]

LabVIEW programs are called Virtual Instruments, or VIs for short. It is different from text-based programming languages (e.g., C, Fortran) in that LabVIEW uses a patented data-flow programming model that frees one from the linear architecture of text-based languages. LabVIEW uses a graphical programming language, known as G programming language, to create programs based on graphical symbols or objects, each symbolizing a programming action. Each of these graphical symbols or objects used to construct the G programs is easily identified by inspection. In LabVIEW it is the flow of data between objects on a block diagram, and not the sequential lines of text, that determines the execution order. LabVIEW is a multitasking system capable of running multiple execution threads and multiple VIs concurrently.[2]

LabVIEW VIs are modular in design, so any VI can run on its own or be used as part of another VI (subVI). With this modularity, one could design a whole hierarchy of VIs and

subVIs that serve as building blocks in any number of applications. LabVIEW provides the user with the ability to modify, interchange, and combine VIs with ease, as an application needs change. LabVIEW thus helps in developing applications much faster than traditional programming languages with its modular and hierarchical structure.[2]

The LabVIEW GPIB (General Purpose Interface Bus), VISA (Virtual Instrument Software Architecture), VXI, and Serial VI Libraries use National Instruments industry-standard device driver software for complete instrumentation control. One can control instruments with RS-232 serial interfaces or any GPIB instrument connected to a National Instruments IEEE 488.2 interface board. VISA, the I/O software interface endorsed by the VXI plug and play Systems Alliance, supplies easy programming of VXI instruments.[2]

With the LabVIEW Instrument Wizard, one can immediately detect any instrument connected to the computer, including GPIB, VXI, Serial, and computer-based instruments. The wizard installs appropriate instrument drivers and in minutes helps to communicate with the connected instrument. LabVIEW instrument drivers translate instrument capabilities into a set of high-level functions to reduce development time and simplify instrument control by eliminating the need to learn the complex low-level programming protocol for each instrument. Nearly 900 instrument drivers are available free on CD or through the Web on the Instrument Driver Network.[2]

The data acquisition (DAQ) VI library acquires waveforms and generates signals with all National Instruments plug-in and remote data acquisition products. The plug-in boards are ideal for high-speed, direct control applications. LabVIEW also has drivers for I/O devices such as PLCs, data loggers, and single-loop controllers. National Instruments offers signal conditioning and remote DAQ modules that integrate smoothly with LabVIEW. The LabVIEW DAQ Solution Wizard and the DAQ Channel Wizard combine to help generate a complete data acquisition solution step-by-step in order to reduce the development time involved.[2]

4.2 The LabVIEW Control System Architecture or Software Platform:

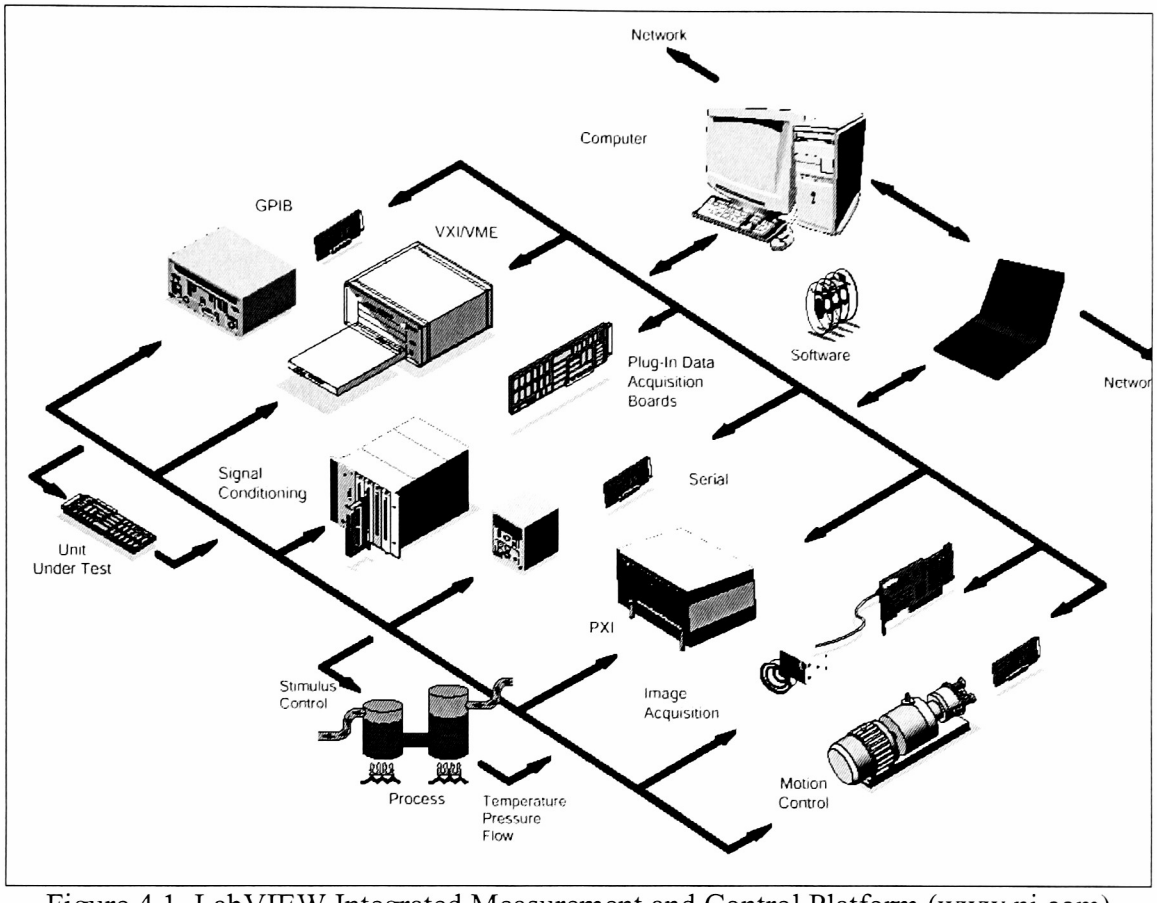


Figure 4.1, LabVIEW Integrated Measurement and Control Platform (www.ni.com)

In theory, control systems can be considered to be a simple set of tasks where we measure a given system, make certain decisions based on the input, send a control signal to adjust the system in order to attain the expected level of operation, and then continue monitoring the system (the set of tasks continue in steps in order to maintain the system under control). [3]

In reality however, accomplishing each of these tasks can grow complex, once we take into account the types of measurements one would need to make in order to get the required inputs, the algorithms and logic needed to make the decisions, the distributed nature of many control systems, the amount of I/O to manage, the speed of the control loop, and so on. If a system is simple and digital, one may be able to find an off-the-shelf

inexpensive hardware controller to do the job. But as the system grows and requires more functionality, one would want to reevaluate such controllers and choose tools that can meet their existing system needs, and then scale to address future changes or technologies as and when they arise.[3]

National Instruments LabVIEW provides a software development environment that is designed to help develop scalable, networked control systems and which caters to any kind of measurement, analysis or algorithm development, and I/O need, from simple to complex applications. The LabVIEW platform of software products provides powerful tools for monitoring, data logging, supervisory control, PID control, fuzzy logic, single point analysis, control prototyping tools, machine vision, and many other applications.[3]

The LabVIEW software platform depicted in figure 4.1 shows how the LabVIEW software residing in the computer acquires process related data from the field or from the real world with the help of various intermediary devices like signal conditioning racks, plug-in data acquisition (DAQ) boards, GPIB, PXI, VXI, and image acquisition cards (for motion control).

The personal computer is in turn connected to a network, where process related data could be shared for monitoring, storage, presentation, or for control purposes. LabVIEW uses a protocol called the DataSocket Transfer Protocol to publish data out to the network or subscribe to a data item available in the network.[4]

4.3 LabVIEW Features:

The basic LabVIEW development environment consists of the Front Panel and the Block Diagram. The front panel is the VI user interface for interactive control and the block diagram consists of the program code that exists in a graphical form (icons, wires, links, etc.). [1]

Front panel and block diagram consists of graphical objects that are the G programming elements. To create the user interface for a VI, the programmer can place the controls and

data displays or indicators (inputs and outputs) for the system on the front panel by choosing objects from the Controls palette, such as numeric displays, knobs, meters, gauges, thermometers, tanks, LEDs, charts, and graphs. Control of the system would be gained at runtime by simply operating the various objects on the front panel, it may be moving a slide, zooming in on a graph, or entering a value from the keyboard.[5] An example front panel that generates a waveform on the basis of a formula is shown in figure 4.2.

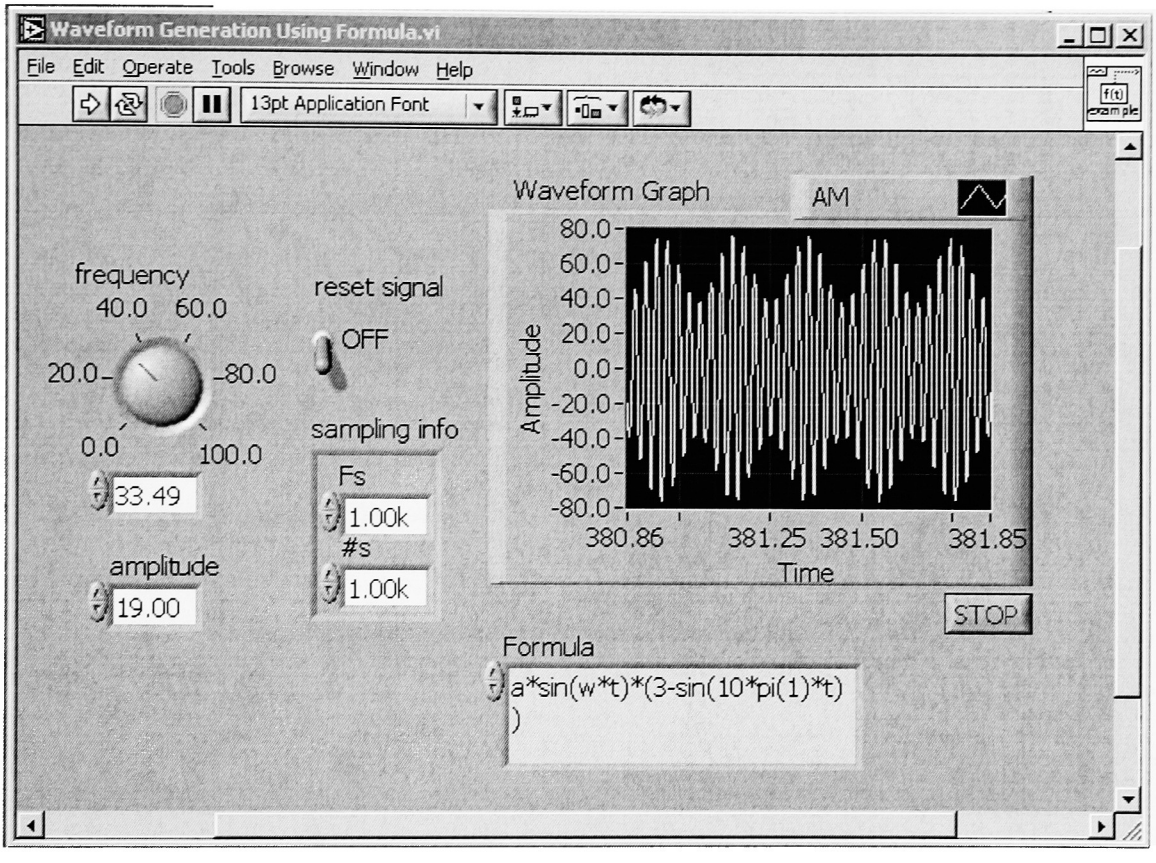


Figure 4.2, Front Panel (www.ni.com)

Block diagrams, the backbone of any LabVIEW program, contain terminals corresponding to the front panel controls and indicators; they are created in the block diagram as we place controls and indicators on the front panel. In addition to the front panel connectivity, the programmer could incorporate the control logic (or the corrective action to be initiated on acquiring an input) into the block diagram.[1] One can construct a block diagram to define the behavior of a VI without having to worry about the many details of conventional programming. One can select objects, or icons, from the

Functions palette and connect them with virtual wires to pass data from one block to the next. These blocks range from simple arithmetic functions to advanced acquisition and analysis routines, to network and file I/O operations. Placing constants, functions, subVIs, structures, and wire that carry data from one object to another into the block diagram provides the required data flow programming.[5] The block diagram for the front panel shown in figure 4.2 is shown in figure 4.3.

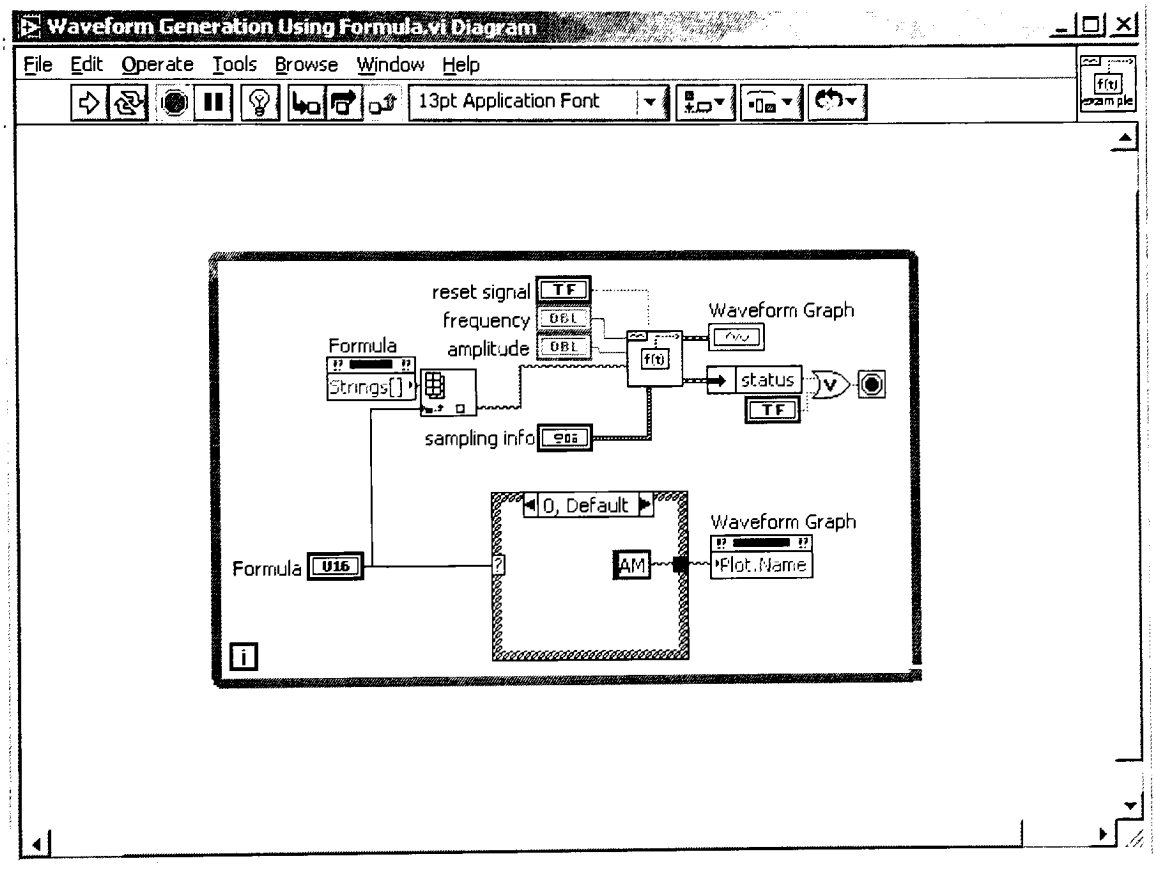


Figure 4.3, Block Diagram (www.ni.com)

In the LabVIEW development environment, the programming code gets compiled as when it is developed pointing the exact error in the program. In many applications, execution speed is a critical consideration. LabVIEW is the only graphical programming system with a compiler that generates optimized code with execution speeds comparable to compiled C programs. To further improve performance, one could analyze and optimize time-critical sections of code with the built-in Profiler. In this way, one could increase productivity with graphical programming without sacrificing execution speed. [2]

LabVIEW provides the programmer with the ability to transfer data acquired from the field, for example temperature readings from a boiler temperature control, into a spreadsheet (Microsoft Excel) for archival. In addition to this feature, it also allows to integrate measurement and automation applications with databases (would need an add-on). These tools deliver high level, easy-to-use functions for integrating local and remote databases quickly and easily into LabVIEW programs. Database tools also allow executing SQL statements from within LabVIEW applications. [4]

4.4 Networking features of LabVIEW in Data Acquisition Environment:

LabVIEW takes advantage of the networking environment to share information or data across the network or to aid distributed execution of tasks. For reporting or monitoring purposes, LabVIEW allows us to publish images of LabVIEW application's user interface to the Web. The built-in LabVIEW Web server publishes images of the VI front panel directly to the web. The Web Publishing Tool allows publishing static and dynamic images of the VI front panel along with additional embellishments, such as titles, text, and borders.[3, 4, 6]

For advanced Internet applications, National Instruments provides us with the Internet Developers Toolkit. The toolkit's Web server goes beyond the built-in LabVIEW Web server by providing more controls over security levels. With the e-mail, FTP, Telnet, and CGI tools that are part of this toolkit, one could automatically send e-mails, send files using FTP, or dynamically decide upon the content to be presented across the Web using Common Gateway Interface (CGI) programs from the LabVIEW application.[4, 6]

Distributed execution allows us to execute various tasks in the measurement system across several computers so that no one machine gets stressed under the heavy load of the required operations, thus improving system performance. Thus one could coordinate measurement operations across many computers into a single system. One common application of distributed execution is remote control. Here one can split the execution such that the measurement system can reside on a remote machine and the controlling portion can reside on a computer in a more comfortable or accessible location. A typical

measurement solution would ideally consist of the acquisition element, the analysis component and the presentation capability all in the same machine, but with distributed execution we could break up these tasks and run them on different machines.[4]

LabVIEW also uses the DataSocket technology that facilitates data sharing over the network. In this arrangement, there are three elements, a publisher, a DataSocket server, and a subscriber. The publisher is the machine, which is the source of the data that is obtained from a measurement or automation application. This data is then published on to a DataSocket server, which in turn functions as a provider of data for the many subscribers or clients whose applications depend on the data provided by it. These three participants could reside on a single machine or could reside on three different machines. DataSocket technology thus enables efficient management of data and prevents duplication required in acquiring the same data for different applications.[4]

4.5 Sharing data across the enterprise with LabVIEW:

Data acquired by LabVIEW will have to be shared with other groups or systems like Distributed Control Systems (DCS), Supervisory Control systems etc for data storing, offline analysis, report generation, and enterprise wide integration. Industry standards such as Ethernet with TCP/IP and OPC (OLE for Process Control) are used more and more in PC-based monitoring systems to seamlessly share the acquired data across the enterprise, from the lab to the factory floor.[4]

Ethernet is quickly moving from the office onto the factory floor because of its low cost, high speed, and multi-protocol ability, clearly becoming the universal network at all levels of the enterprise. Ethernet can be implemented not only at the higher level of the plant hierarchy, but also at the control level to connect distributed I/O to PLCs and data acquisition systems. Ethernet and TCP/IP form a solid framework for networking and Web connectivity. However, to easily share multiple I/O and data acquisition devices with multiple PCs, the application-level protocol plays an important role. The application-level protocol basically defines the common language used to talk to the different devices.

In the industry, standards such as OPC and HTTP are rapidly gaining consensus. OPC provides a whole new level of interoperability between automation software and hardware, removing the obstacles created by proprietary industrial device interfaces and drivers. It defines a standard for a common interface to communicate with a variety of industrial devices and communication protocols. OPC is the answer to multi-vendor connectivity problems, which are specifically critical in monitoring applications.[4]

4.6 LabVIEW Datalogging and Supervisory Control for Increased Productivity:

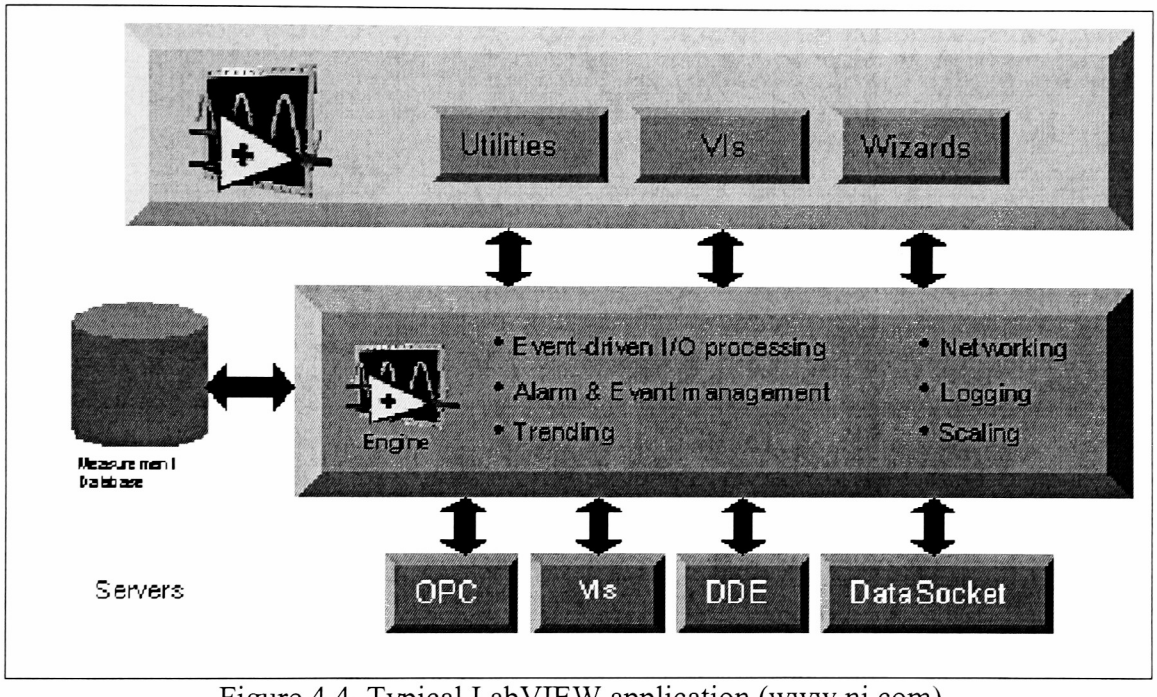


Figure 4.4, Typical LabVIEW application (www.ni.com)

In order to increase the usability and effectiveness of LabVIEW in a setup that contains a wide range of application requirements (data logging, networking, Web-based application) and data sources or devices that are supplied by different vendors, National Instruments provides an add-on tool called LabVIEW Datalogging and Supervisory Control Module. The LabVIEW Datalogging and Supervisory Control Module provides built-in tools to perform traditional monitoring tasks, such as alarm handling, historical logging and viewing, data management, and networking, thus yielding a significant increase in productivity. Also, with easy networking, distributed logging, built-in security, and OPC connectivity, the LabVIEW Datalogging and Supervisory Control

module provides tremendous ease of use to get a monitoring system up and running quickly; and by adding full OPC client and server capabilities to a LabVIEW application, one could communicate with any OPC server available on the market today.[4]

This LabVIEW add-on module also adds an event-driven engine to the LabVIEW application. This engine runs in the background and maintains the real-time database, logs historical data, processes alarm information, and communicates with device servers.

The event-driven architecture of the engine greatly improves the performance of a system by freeing up CPU time until an event occurs. An event is generated when a data point changes, a switch is toggled, or an alarm is generated. When the event occurs, the system is notified and executes the specified functionality. The clear advantage of such architecture is that the system has more capacity for growth without performance degradation.[4]

References:

1. Bishop, R.H – Learning with LabVIEW. California: Addison Wesely Longman, 1999.
2. National Instruments Corporation. "LabVIEW Basics" 1999, < http://www.tinkersguild.com/sample/SponsorAds/NatInstruments/labview_basics.htm>
3. National Instruments Corporation. " Developing Scalable Networked Monitoring and Control Systems with LabVIEW" 2002, < http://zone.ni.com/devzone/conceptd.nsf/webmain/199C3FB9456A16D986256B5200678917?opendocument>
4. National Instruments Corporation. "LabVIEW – Proven Productivity, An In – Depth Look at Graphical Programming" Texas, September 2000, Part Number 350150F-01
5. National Instruments Corporation. "LabVIEW Graphical Programming" 1999, < http://www.tinkersguild.com/sample/SponsorAds/NatInstruments/lv_program.htm>
6. Shirer, D.L., "LabVIEW 6i Adds Internet Features to Data Acquisition Environment" Computing in Science & Engineering, Volume: 3 Issue: 4, July-Aug. 2001, Page(s): 8 –11

5. LabVIEW DataSocket Technology

5.1 Introduction: [1]

A technology resident in LabVIEW that facilitates data sharing over the network is DataSocket. DataSocket is extremely useful because it simplifies streaming of data between different applications on one computer or between hosts of computers in a network. Although a variety of technologies exist today to share data between applications, such as TCP/IP (Transmission Control Protocol/ Internet Protocol) and DDE (Dynamic Data Exchange), most of these tools are not targeted specifically and optimized for streaming data like DataSocket. DataSocket uses an enhanced data format for exchanging measurement data, as well as attributes of the data. Data attributes might include information such as an acquisition rate, test operator name, timestamp, quality of data, etc.

DataSocket applications require three participants a ***Publisher***, the ***DataSocket Server***, and a ***Subscriber***. Both the publishing and subscribing applications are "clients" of the DataSocket Server. These three participants can reside on the same machine or they can all run on completely different machines. Examples of each of these configurations will be explained later in the chapter. The ability to run them on separate machines improves performance and provides security by isolating network connections from the measurement application.

5.2 Publisher:

A publishing application uses the DataSocket API to write data to the DataSocket Server. The publisher writes the data to an identifier, or tag, formatted like a Uniform Resource Locater (URL). URL is a standard that uses a descriptive string to identify the source or connection for a data transfer used with DataSocket, FTP, HTTP, OPC, and file I/O. The URL includes the communication protocol (scheme), name or identifier of the computer being accessed, and scheme-specific information identifying a file, Web page, server, or, when using DataSocket, the data item. DataSocket URLs are constructed by simply adding **dstp://** before the destination or the computer name where you wish to publish the

data. An example of one such DataSocket connection that uses the DataSocket protocol is **dstp://imert09.cims.rit.edu/signal**. The DataSocket Transfer Protocol (**dstp://**) is based on TCP/IP to exchange data between two applications using DataSocket clients. Data is passed through a DataSocket Server between applications.

5.3 DataSocket Server: [2]

The DataSocket server can broadcast live measurement data at high rates across a network or across the Internet to several remote clients concurrently. It simplifies the measuring application by automatically managing the connection to clients. DataSocket server is only available on Windows platform.

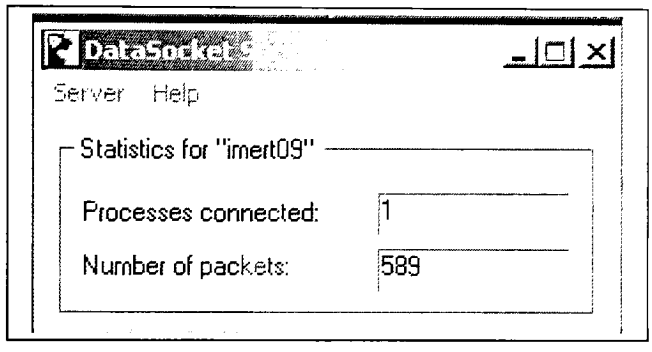


Figure 5.1 DataSocket Server (LabVIEW6i Development System)

The DataSocket Server accepts and stores information from the data sources and relays it to other data targets (or subscribers). One can launch the DataSocket Server from **Start»Programs»National Instruments DataSocket»DataSocket Server**.

The default configuration of the machine allows the local machine to create new data items and to change the data in these data items. Remote machines can connect to the server and read items. To create an item, run a client that uses a DataSocket control to write items (publish) to the server. When a DataSocket writes data to a data server, the server automatically creates a new entry and stores the value in that entry.

If the reading client (subscriber) connects to the server, it gets updates only after calling the Update method on the DataSocket. The data received by the subscriber from the

server gets automatically updated when another client (publisher) changes the value. The data does not get updated until a writing client connects to the same data item and actually writes a value to it.

5.4 DataSocket Server Manager: [2]

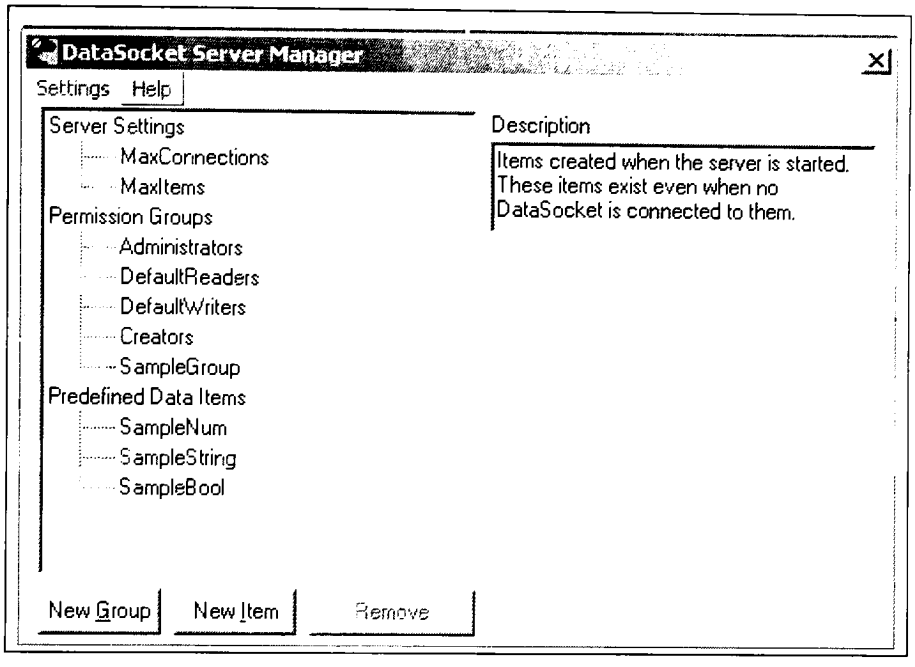


Figure 5.2, DataSocket Server Manager (LabVIEW6i Development System)

The DataSocket Server Manager offers the ability to configure and manage the security of applications by allowing the user to set options such as access permission, number of connections, and predefined data items stored in the DataSocket Server. One could launch the DataSocket Server from **Start » Programs » National Instruments » Measurement Studio » Utilities » DataSocket Server Manager**.

By default, only programs running on the same computer as the server can create items or write to items, whereas applications on the same computer or other computers can read items. To allow an application running on a different machine to connect to the DataSocket Server, one has to just increase the number of connections a DataSocket Server would allow.

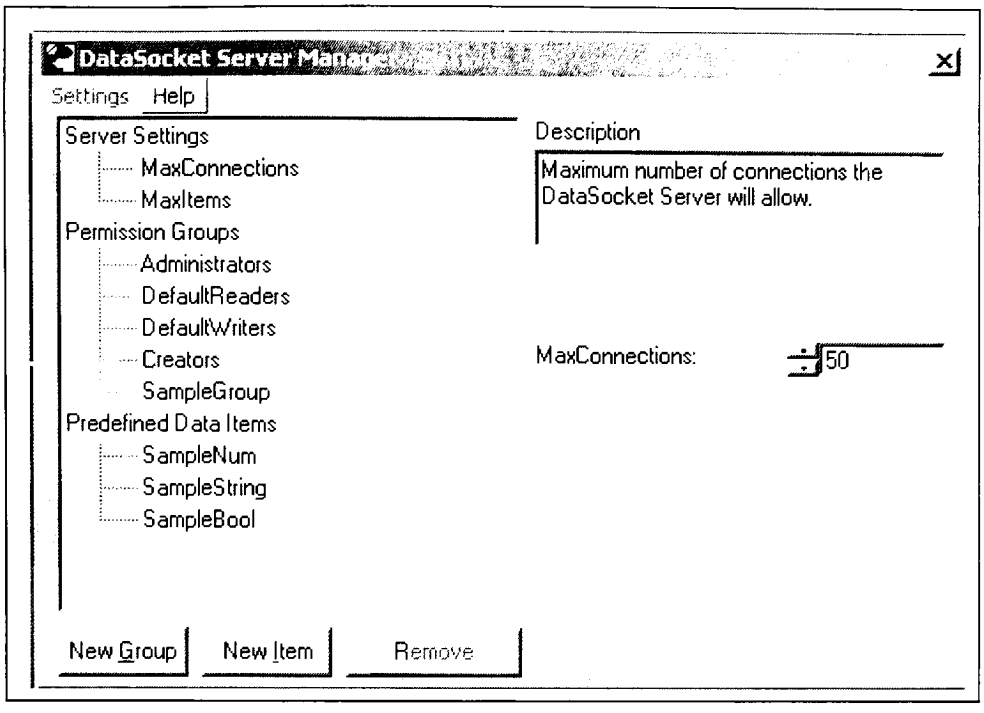


Figure 5.3, DataSocket Server Configuration (LabVIEW6i Development System)

Items exist only as long as there is at least one DataSocket client connected to read or write the items value. When no connections to the item remain, the DataSocket server releases the item and its value.

One can specify items that should be automatically created and given an initial value when the server is started. The DataSocket server never releases predefined items, so their values exist even when no DataSocket client is connected to them. Predefined items can have special read and write access groups so different machines can have different access to different items. Changes that are made to the server take affect the next time the DataSocket server is launched.

5.5 Single Machine Application:

Here we look at a Single Machine Application where all three participants, namely- Publisher, DataSocket server, and the Subscriber reside in the same machine. We pick a computer IMERT09 for the application.

Firstly, launch the DataSocket Server from **Start»Programs»National Instruments DataSocket»DataSocket Server**. Create a VI to acquire random numbers produced by a Random Number Generator, multiply the number by 100 and display the output with the help of a waveform chart.

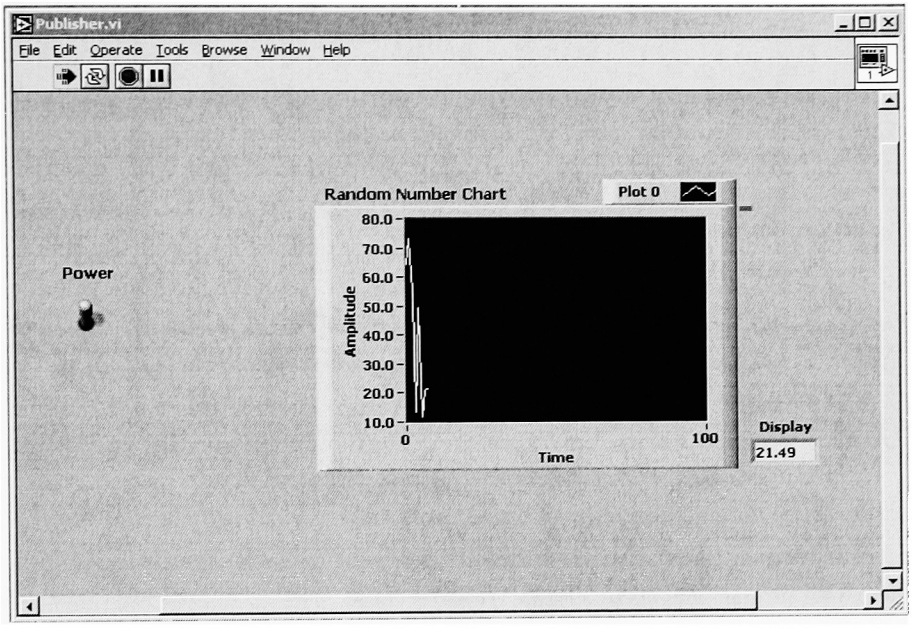


Figure 5.4, Random Generator VI

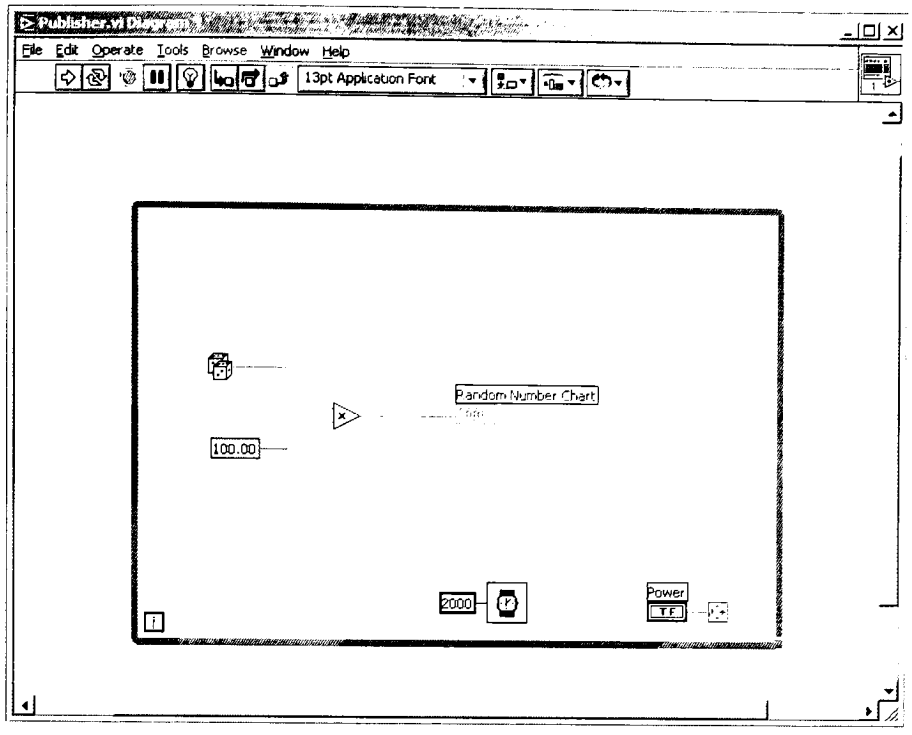


Figure 5.5, Random Generator Diagram

Right Click on the waveform graph and choose **Data Operations>>Data Socket Connection...** from the shortcut menu. A DataSocket Connection will appear as shown in figure 5.6.

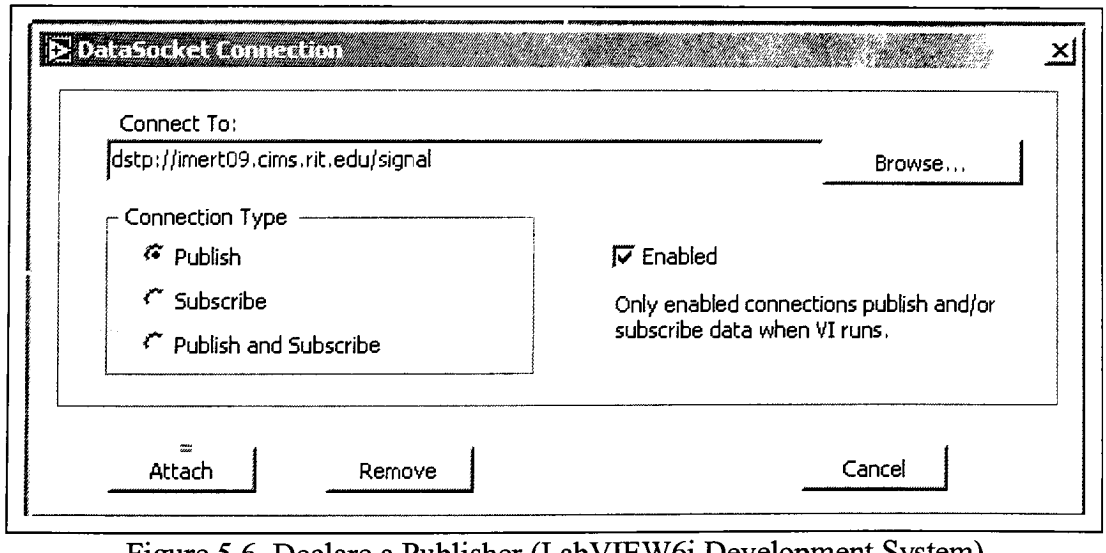


Figure 5.6, Declare a Publisher (LabVIEW6i Development System)

In the DataSocket Connection window, enter **dstp://imert09.cims.rit.edu/signal** below the **Connect To:** heading, click on the **Publish** radio button, and press the **Attach** button. This will close the DataSocket Connection window and setup a publishing connection named **Signal** between the waveform chart and the DataSocket server.

Now open a new VI by pressing **Ctrl+N**. Place a Waveform Graph on the Front Panel (Graph subpalette of the Controls palette). Right click on the Waveform Graph and choose **Data Operations>>Data Socket Connection**. In the DataSocket Connection window that appears, click on the **Browse** button and select **dstp://imert09.cims.rit.edu/signal** from the list.

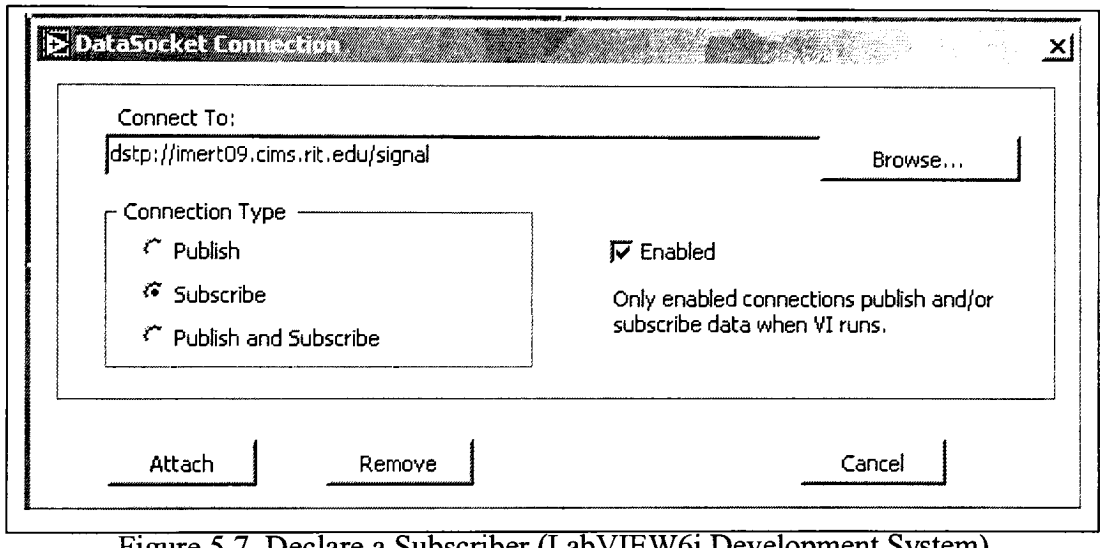


Figure 5.7, Declare a Subscriber (LabVIEW6i Development System)

Click the **Subscribe** radio button and press **Attach**. The DataSocket Connection window will then disappear.

Press the **Continuous Run** button and one could observe the data displayed in the Waveform graph of the new VI. One should notice that no programming was done to subscribe to the data.

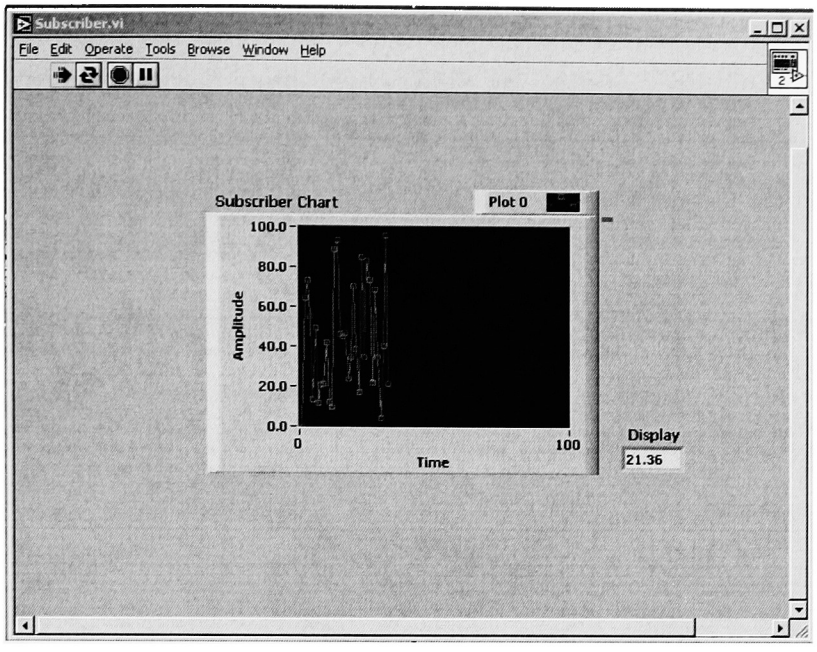


Figure 5.8, Subscriber VI

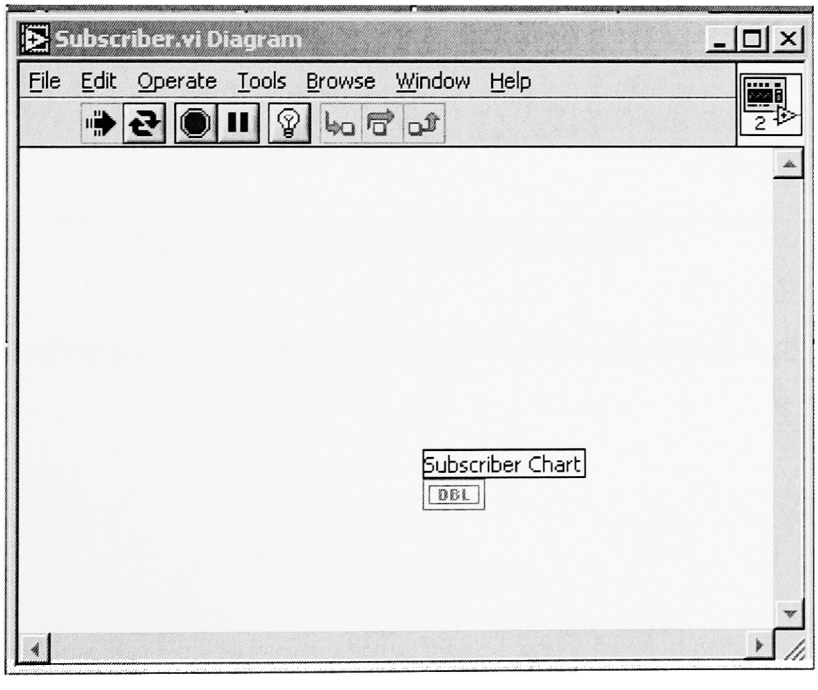


Figure 5.9, Subscriber Block Diagram

In the VI shown in figure 5.9, there is a single orange **Subscriber Chart** terminal on the Block Diagram with nothing wired to it. The DataSocket Connection feature allows publishing and subscribing of measurements with no programming.

5.6 Multiple Machine Application:

We look at a multi-machine application where all the three participants, namely- Publisher, DataSocket server, and the Subscriber reside in three different machines. We pick computers IMERT09, IMERT14, and IMERT12 for the application. The DataSocket scheme for the above configuration is as shown in figure 5.10.

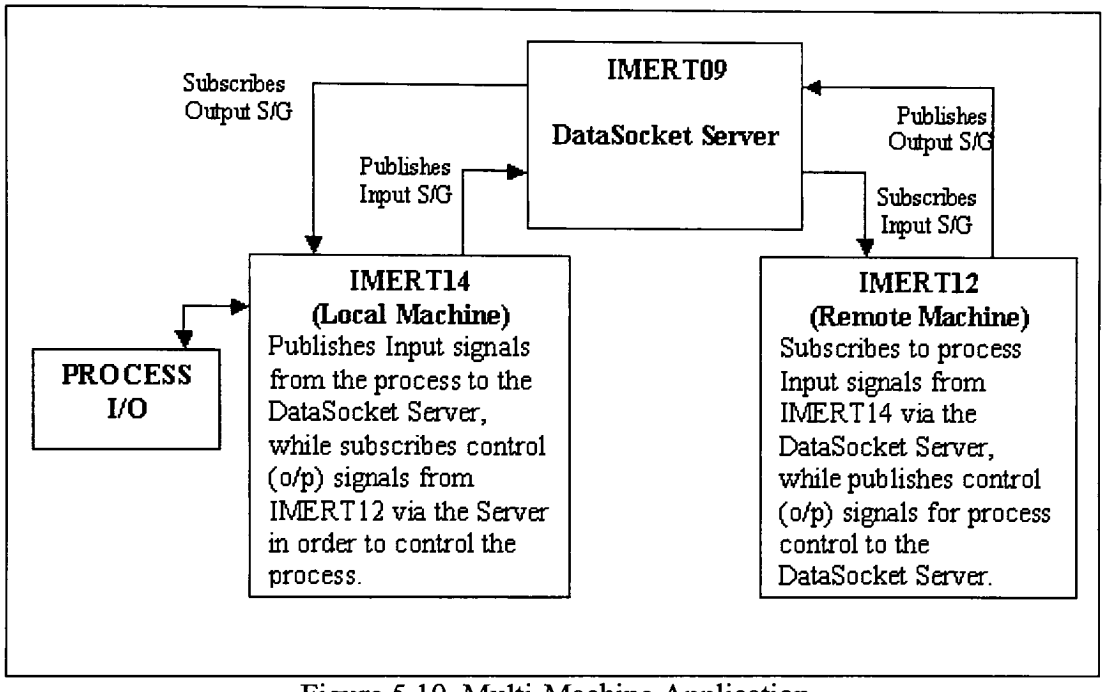


Figure 5.10, Multi-Machine Application

The Front panel of the program running in IMERT14 is shown in Figure 5.11

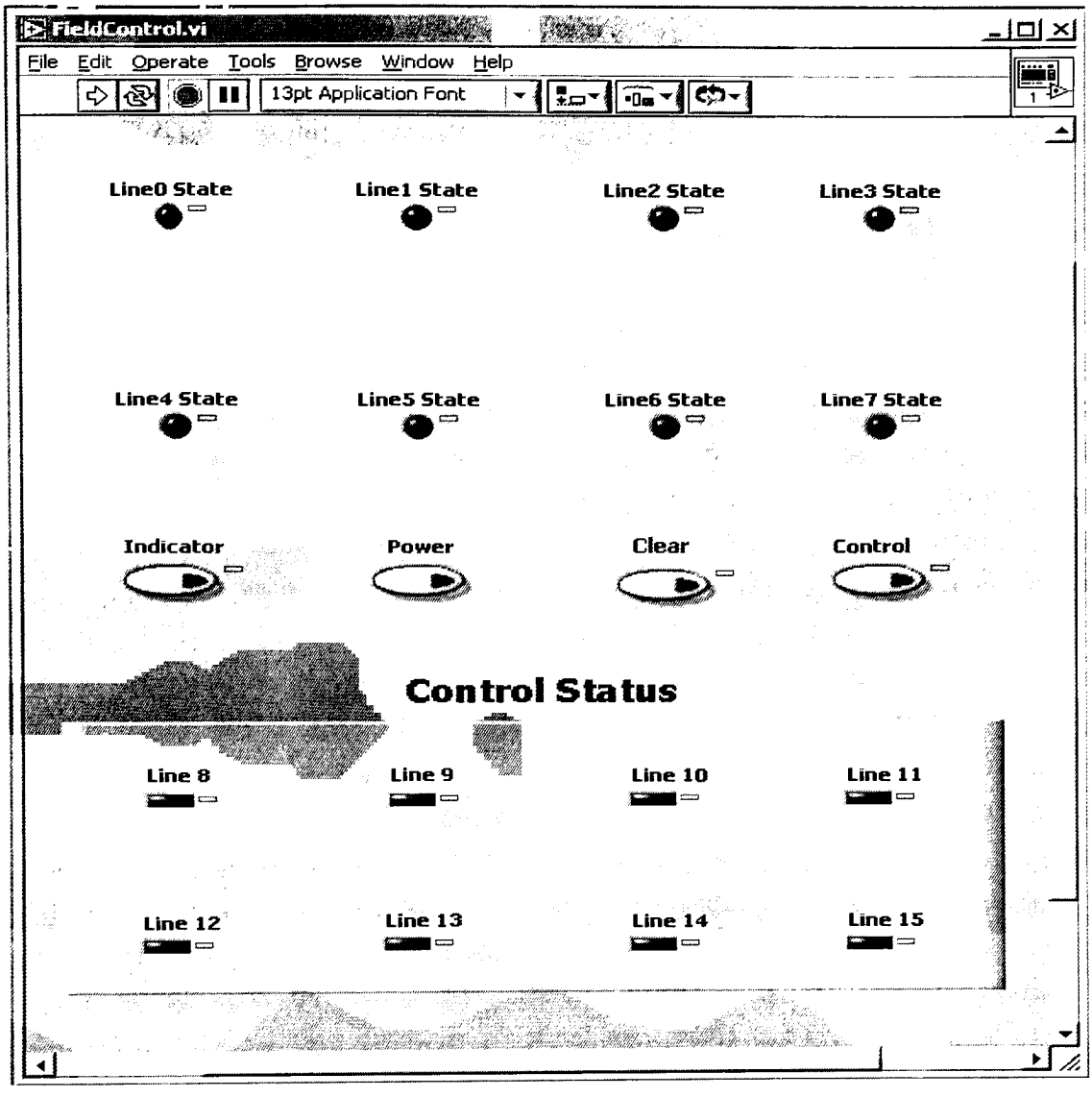


Figure 5.11, Field Control VI on IMERT14

The eight LEDs named from **Line0 State** to **Line7 State** are the Input indicators that monitor the input from the process and publish the state (High or Low- Boolean Format) to the DataSocket Server on IMERT09, as and when there is a change in each of the input states.

The eight **Control Status** indicators named **Line8** to **Line15** are the output lines that subscribe to values from the DataSocket Server in the form of Boolean data; the value that is obtained from the DataSocket Server is used to manipulate the process, e.g., turn

the motor clockwise, turn the motor counter clockwise or stop the motor. **Line8** to **Line15** are the reading clients whose values that they subscribe to gets updated only after the writing client (controls on the VI running on IMERT12) publishes the new values to the DataSocket Server.

The Front panel of the program running in IMERT12 is shown in Figure 5.12

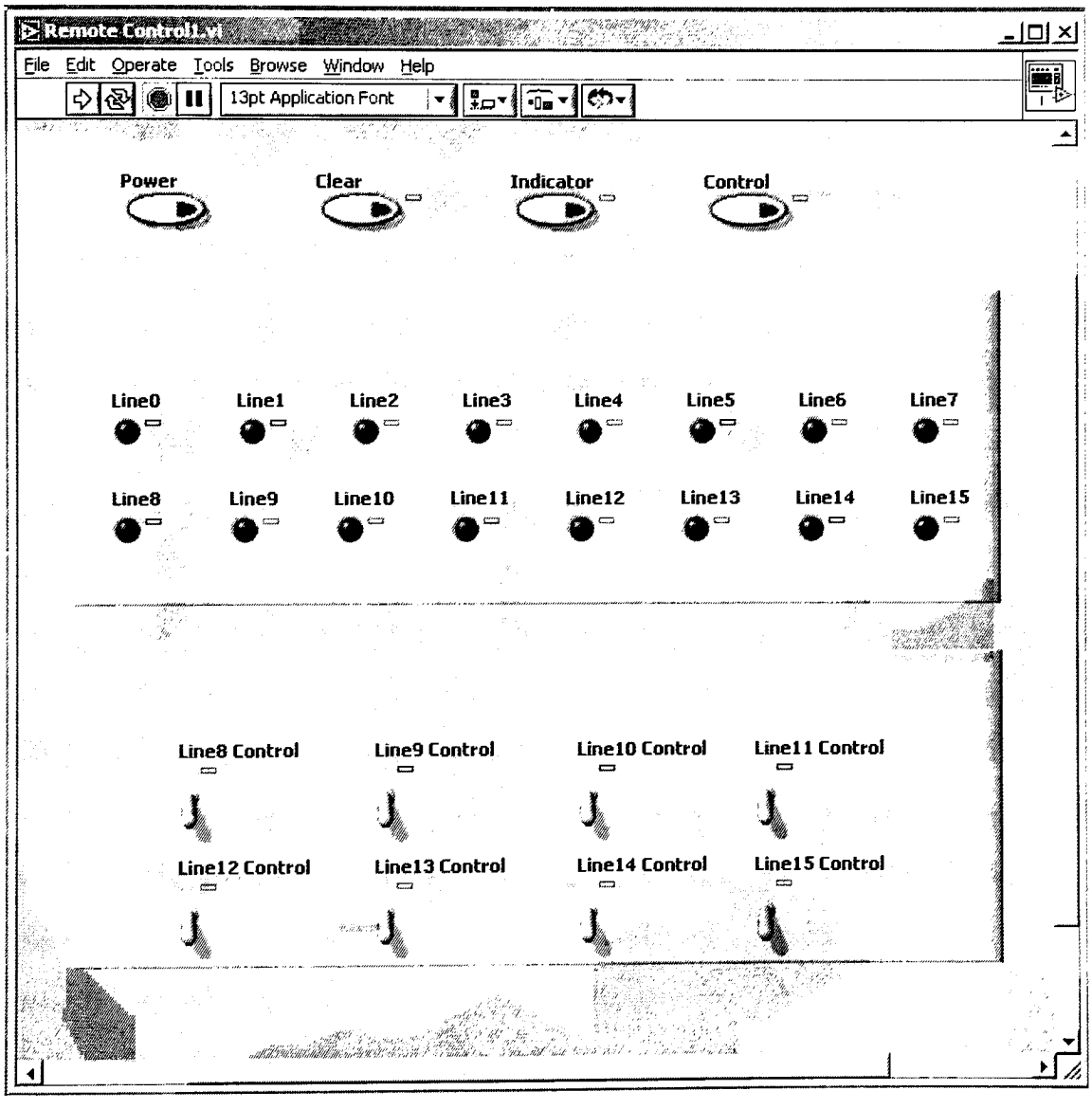


Figure 5.12, Remote Control VI on IMERT12

In the VI shown in figure 5.12 the eight indicators from **Line0** to **Line7** are input indicators that monitor the input from the process by subscribing to the individual states from the DataSocket Server resident on IMERT09. Indicators **Line0** to **Line7** are the reading clients whose values that they subscribe to gets updated only after the writing client (process inputs read by the VI running on IMERT14) publishes the new values to the DataSocket Server. The other eight indicators are merely subscribing from the DataSocket Server the data published by the toggle switch controls **Line8 Control** to **Line15 Control** resident in the local VI.

The eight toggle switch controls named **Line8Control** to **Line15 Control** are the controls that publish its state to the DataSocket Server. This value is subscribed by the eight toggle switch controls of the VI running on IMERT14 to cause a change in the process that it controls.

References:

1. National Instruments Corporation. "LabVIEW – Proven Productivity, An In – Depth Look at Graphical Programming" Texas, September 2000, Part Number 350150F-01
2. LabVIEW 6i Online Help – National Instruments Corporation

6. OLE for Process Control

6.1 Introduction: [1]

Industrial automation users are entering a new era in which seamlessly integrated, multi-vendor control systems will become a reality, and proprietary software and hardware interfaces will become a thing of the past. Over the last two decades, the industrial automation market has seen a proliferation of proprietary interface standards. Literally hundreds of different and incompatible proprietary interface standards developed by suppliers are required to communicate with today's automation systems and devices. It is only by working through this maze of interfaces that today's point-based solutions, such as field devices, control systems, and business systems, can be turned into an integrated plant-wide or mill-wide manufacturing system.

Lack of software standards limit the effective use of real-time information from today's control systems. Difficulties in integrating the many required software interfaces put a bottleneck on the overall system capability to provide information on demand. In addition, a lack of standards often means a lack of choice for users. Creating a multi-vendor system, in which a company can choose and apply the best products and systems for any given application often requires a willingness to invest a significant amount of time and money on system integration. Much effort is required to ensure that the systems, which are purchased, can share information and interoperate with other automation and business systems in the plant or factory.

New software technology is bringing about dramatic changes in the automation industry. These technological advances are bringing into focus the reality of plug and play software. The industrial automation industry is embracing a new breed of software technology that emphasizes flexibility, openness, and ease of integration. This new technology provides enterprise-wide tools that are scalable, platform-independent, based on object-oriented graphical software development environments, and work in distributed client/server architecture. Technologies such as Microsoft's ActiveX, based on OLE/COM (Object Linking and Embedding/Component Object Model), Sun's Java

language, and Internet/Intranets are all new technologies that are working towards the goal of open standards and interoperability. Suppliers and users must work together to define how to leverage and use these technologies before they effectively create manufacturing and enterprise-wide solutions[1].

6.2 OLE for Process Control (OPC): [1]

A promising new standard for the industrial automation market is OLE for Process Control, or OPC. An organization, the *OPC Foundation*, has been created to establish guidelines by which OPC will provide multi-vendor interoperability throughout the plant floor. The OPC Foundation's charter is to establish an open connectivity standard based upon Microsoft's OLE/COM (component object model). The OPC Foundation put forth the goal of developing a single client/server specification that would allow any vendor to develop software and applications that could share data in a fast, robust fashion, and do it in a way that would eliminate the proprietary schemes that forced these same vendors to duplicate development efforts. The OPC Foundation developed the first specification, called the Data Access Specification 1.0a, which was released in early 1996. Using this specification, vendors were able to quickly develop client/server software. OLE/COM is an important part of Microsoft's client/server distributed computing strategy. With its set of interfaces, an end-user can interoperate and communicate between distributed components. However, to effectively use OLE as part of an integration framework, manufacturers must agree on OLE standards. OPC defines a set of OLE/COM interfaces, properties, and methods that extend this technology, making it useful in process and manufacturing automation applications. This initiative has two main benefits. First, OPC will enable control and business applications running on distributed, heterogeneous platforms to integrate at the object level. Second, OPC eliminates the need for different proprietary hardware and software device communication drivers. OPC will provide plug-and-play communication and interoperability between field devices, control systems, and enterprise-wide business applications.

6.2.1 What is COM-DCOM? [2]

COM or Component Object Model is an outgrowth of the object-oriented paradigm and is a specification that is based on a binary standard for reuse through interfaces. COM is all about a binary level of interoperability between clients and COM objects. COM defines a standard for the component interoperability, which makes it able to interact with one another without knowing the internal details of the language in which it is written in.

DCOM or Distributed Component Object Model, which is an extension of Component Object Model, has proven to scale well on the internet, intranet and beyond. It is a set of Microsoft concepts and program interfaces in which client program objects can request services from server program objects on other computers in a network.

Utilizing OLE/DCOM technology for developing software standards to communicate with industrial devices led to the OLE for Process Control (OPC) specifications.

6.2.2 OPC Specifications: [3]

OPC (OLE for Process Control) defines a set of interfaces, based on OLE/COM and DCOM technology, for truly open software application interoperability between automation/control applications, field systems/devices and business/office applications (See Figure 6.1). OPC, which is managed by the OPC Foundation, has more than 200 companies and institutes as members.

The foundation has released two sets of interface specifications:

- OPC Data Access (V2.0, October 1998)
- OPC Alarms and Events (V1.0, December 1998)

The others in preparation are:

- OPC Historical Data
- OPC Security
- OPC Batch

Of the two specifications listed above, OPC Data Access is the only one actually supported by the industry.

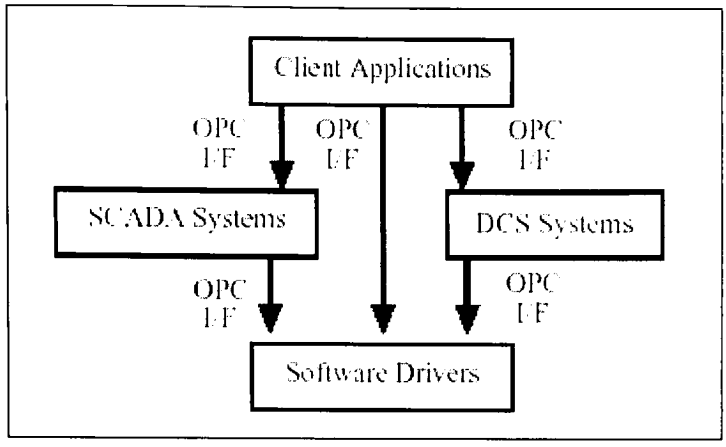


Figure 6.1, OPC Applications (R. Barillere, et. al.)

6.2.3 OPC Data Access: [4]

The primary intent of the Data Access effort, the first of the OPC efforts, was to solve "the I/O Driver problem". Figure 6.2 shows "the I/O Driver Problem":

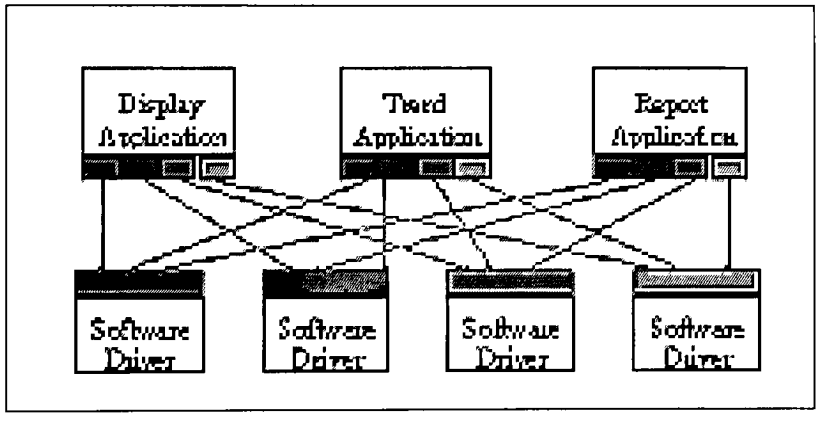


Figure 6.2, The I/O Driver Problem (www.intellution.com)

From figure 6.2, we see that each software driver for a new device requires a unique, device-specific chunk of code to talk to the device or network. The need for this device specific code is not a problem OPC is trying to address. Rather, this is something that data acquisition networks such as Devicenet, Profibus, and Lonworks, and a host of others are trying to address.

The problem that OPC deals with is that each of these software drivers also presents a different software interface to the calling application. As a result, each application also requires a unique chunk of code for each driver, device or network it wants to connect to. This obviously makes it difficult to reuse applications with different sets of process interface equipment.

OPC solves this problem by creating a "software bus" as shown in figure 6.3. Applications only need to know how to get data from OPC data sources. As a result, applications would be much simpler, smaller and easier to use. Device Drivers (servers) now only needs to know how to provide data in a single format (OPC Server).

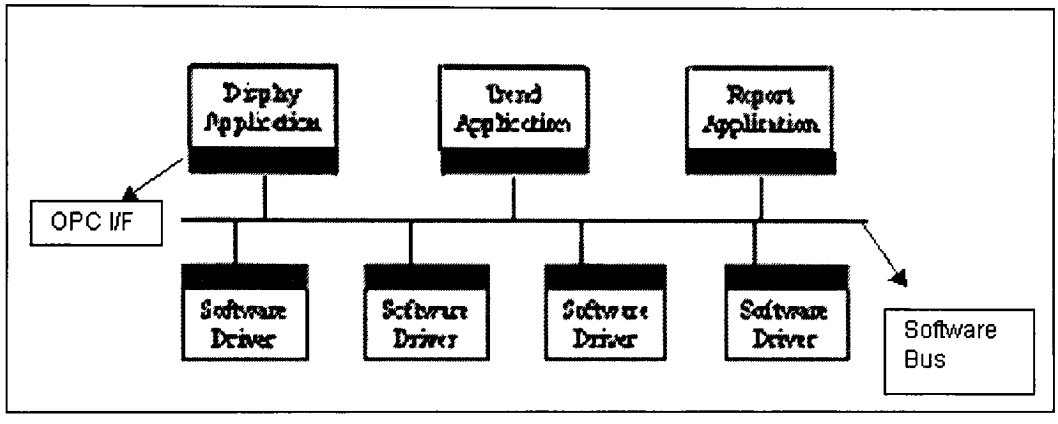


Figure 6.3, OPC Implementation (www.intellution.com)

Although the main problem OPC solves is the "I/O Driver", we see from figure 6.1 that OPC is flexible enough to be used not only between Devices and SCADA, SoftPLC or DCS systems, but also between the SCADA SoftPLC or DCS engines and the high level applications such as HMI, Trends, Reports, etc. which use their data. In this architecture, the SCADA system, the DCS system or the SOFTLOGIC system really just represents a "smarter" device.

As a result of this focus, the Data Access interface is intended to grab real time snapshots of process variables such as temperature, pressure, flow rate, position, speed, weight, contact closure, etc. It does this by allowing the application to define one or more "groups" in order to monitor the items in which it is interested. The client either conducts

synchronous reads of items in the group or subscribes to the group on an exception basis. For ease of use, a browser interface is supported which allows the client to obtain a list of the items in the underlying system for which data is available.

6.2.4 OPC Alarms and Events: [5]

The primary intent of the Alarms and Events effort is to provide an interface to allow applications such as alarm loggers or MMI packages to "subscribe" to various classes of events using various filtering capabilities (e.g., by class and/or plant area). The design is intended to support a very flexible set of events such as process alarms, status changes, operator tracking messages, operator manual action requests, batch completion messages, operator annotations, etc. Some events (alarm notification or operator action) may require acknowledgement while others (tracking messages) may not. Again, since this is an interface definition, the specification does not dictate how the server generates the messages or how the client makes use of them. Unlike Data Access, the client does not explicitly list the items to which it wants to subscribe (such as the thousands of individual alarm conditions which might be checked by a large SCADA system). Instead it subscribes to all events, which fit a certain specified list of criteria. The design is also intended to support multi tier implementations.

In this case, various browse interfaces are provided which allow the client to determine the filter criteria (e.g., plant areas) supported by the particular server.

6.2.5 OPC Historical Data: [5]

The primary intent of the Historical effort is to provide an interface to allow for the retrieval of data from the historical archive for use by various applications such as MMI, Report Generation, Analysis, etc. The data provided by the historical interface consists of historical records and optionally, various "post processed" values such as min, max, average, etc. As with Data Access, the clients provide lists of items to the server in the form of groups. In this case, rather than grabbing a real time snapshot, the client also provides the time span and number of samples needed. In addition to the data retrieval interface, this specification will provide for maintenance or editing interface.

Also, as for Data Access, a browse interface allows the client to determine the items for which data is available in the historical log.

6.2.6 OPC Security: [5]

The security effort is expected to investigate the need for systems to control access to various process parameters based on a user ID. It is unclear at this time exactly what the best solution to this problem will be.

6.2.7 OPC Batch: [5]

The batch effort is expected to investigate the use of existing or additional OPC interfaces for access to various types of batch data including recipe (data that flows into a PLC) data as well as real time data. It is possible (but not certain) that the current OPC Data Access interface, perhaps with the addition of a specific naming convention might be able to handle much of this task.

6.2.8 General OPC Architecture and Components: [6]

OPC specifications always contain two sets of interfaces, Custom Interfaces and Automation interfaces. This is shown in figure 6.4.

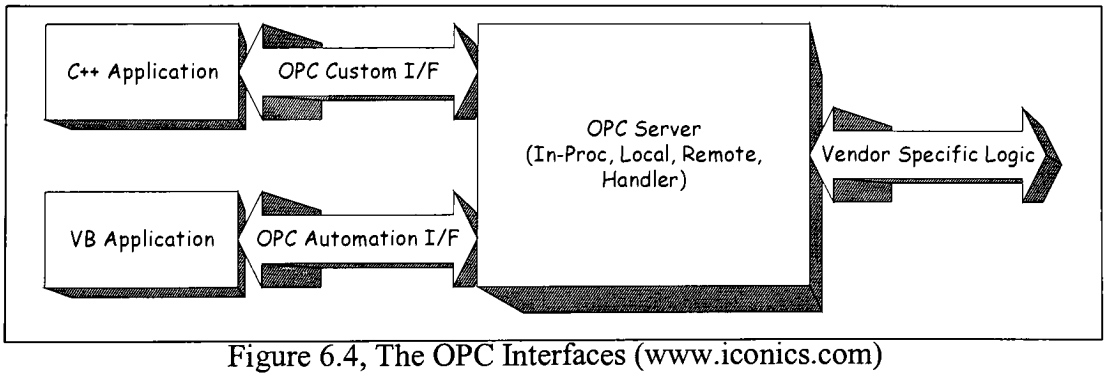


Figure 6.4, The OPC Interfaces (www.iconics.com)

The OPC Specification specifies COM interfaces (what the interfaces are), not the implementation (not the how of the implementation) of those interfaces. It specifies the behavior that the interfaces are expected to provide to the client applications that use them.

Like all COM implementations, the architecture of OPC is a client-server model where the OPC Server component provides an interface to the OPC objects and manages them.

There are several unique considerations in implementing an OPC Server. The main issue is the frequency of data transfer over non-sharable communications paths to physical devices or other databases. Thus, we expect that OPC Servers will either be a local or remote EXE, which includes code that is responsible for efficient data collection from a physical device or a database.

An OPC client application communicates to an OPC server through the specified custom and automation interfaces. OPC servers must implement the custom interface, and optionally may implement the automation interface. In some cases the OPC Foundation provides a standard automation interface wrapper. This "wrapperDLL" can be used for any vendor-specific custom-server.

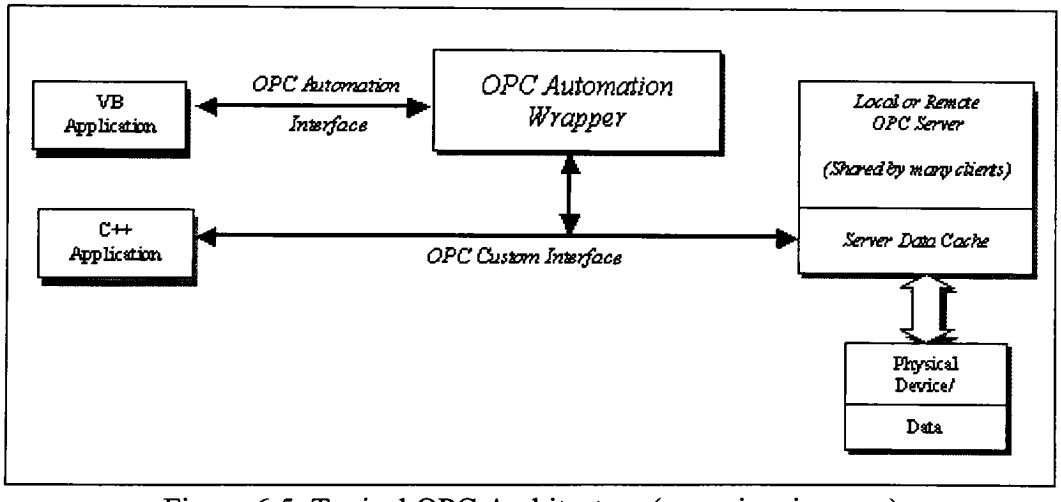


Figure 6.5, Typical OPC Architecture (www.iconics.com)

6.3 OPC Data Access Specification: [4, 7]

OPC Data Access is based on Microsoft's COM technology in order to take advantage of its common 'plumbing' required to connect the clients and servers including DCOM which allows creation of network based applications with a minimum of development effort. As noted earlier OPC is flexible enough to be used at different layers within the system architecture.

OPC efficiency is insured by the fact that

- OPC can be exception based
- Unlike DDE, OPC can pack 100s or 1000s of data items into each transaction

Performance is guaranteed both by the support for multiple data items per transaction and by the fact that COM and thus OPC servers can be created in several models

- Inprocess
- Local
- Remote

Inprocess servers incur zero overheads. In this setup millions of transactions per second are possible. Local servers allow 1000s of transactions per second. Even remote servers can manage 100s of transactions per second. And again, unlike DDE, each transaction can include many data items.

6.3.1 Problems not solved:

There are a number of problems that OPC data access does not solve. At the same time OPC does not prevent the implementer from solving these problems. Some of these problems or shortcomings are:

Global Naming: One cannot give a command of 'find FIC101' in order to find a device connected to a particular OPC Server. One will have to know what software server and what physical node to look on.

Alarms and Events: These issues are taken care of another effort altogether.

Security: It was thought of earlier that solving the I/O driver problem should be the primary goal, due to which security was not considered as an issue. The SCADA engine was assumed to be a trusted component.

6.3.2 Assumptions about the Architecture:

Certain assumptions made by the Data Access Specification are:

- Each OPC Server such as Data Access is a separate object.
- The Data Access Server provides a window into the 'Existing Data' from the device; it is not a configuration system.
- Data is accessed by Name (a string) which will generally be vendor or hardware specific
- Data for lists of items can be read explicitly (polled) or subscriptions can be created.

6.3.3 Assumptions about the Applications:

Data Access Specification has made certain assumptions for the Application as well. They are:

- Applications are interested in a subset of the Data Items (tags) available within the underlying Control sub-system.
- Applications are interested in the many subsets of Data Items at different times and may have variable requirements for response and resolution.
- Applications want to be independent of the data structures (or objects) used by the sub-systems (wanting symbolic access of the data).

A fundamental assumption here, is that there is some sort of control or monitoring engine running independently of anything OPC is doing and that OPC is just a window into this engine's data.

6.4 The Logical Object Model: [7]

The Data Access Specification Object Model is as shown in the figure 6.6

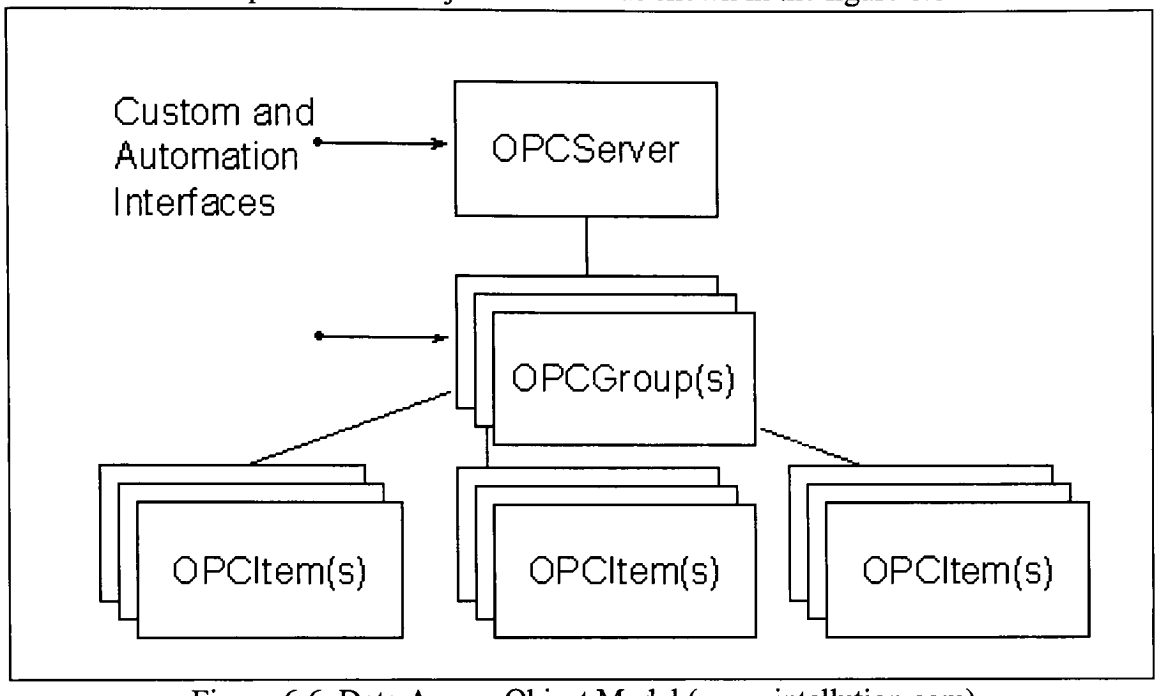


Figure 6.6, Data Access Object Model (www.intellution.com)

At a high level, an OPC Data Access Server is comprised of several objects: the server, the group, and the item, as shown in figure 6.6. The OPC server object is the COM object to which the application first connects. These maintain information about the server and serve as a container for OPC group objects.

OPC group objects are created dynamically by the applications to hold lists of Tags and attributes (which OPC refers to as Items). OPC Groups maintain information about itself and provides the mechanism for containing and logically organizing OPC items. The OPC Groups provide a way for clients to organize data. For example, an HMI might create a Group for each open picture or a report package might create a group to access data for each report.

The contents of the Group and item collections may vary over time based on the needs of the applications. Different Groups can be used by different parts of the application. Data can be read and written. Exception based connections can also be created between the

client and the items in the group and can be enabled and disabled as needed. An OPC client can configure the rate that an OPC server should provide the data changes to the OPC client.

There are two types of groups, public and local (or "private"). Public is for sharing across multiple clients, local is local to a client. There are also specific optional interfaces for the public groups. Within each Group the client can define one or more OPC Items.

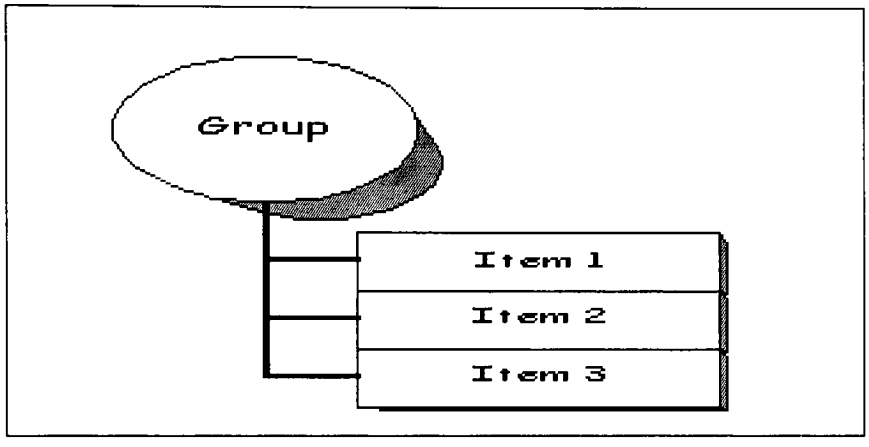


Figure 6.7, Group/Item Relationship (www.intellution.com)

The OPC Items represent connections to data sources within the server. An Item would have connection to a single process value or setpoint. An OPC Item, from the custom interface perspective, is not accessible as an object by an OPC Client. Therefore, there is no external interface defined for an OPC Item. All access to OPC Items is via an OPC Group object that "contains" the OPC item, or simply where the OPC Item is defined. Associated with each item are a Value, Quality and Time Stamp. The value is in the form of a VARIANT, and the Quality is similar to that specified by Fieldbus.

However one should note that, the items are not the data sources they are just connections to the data sources. For example, the tags in a DCS system exist regardless of whether an OPC client is currently accessing them. The OPC Item should be thought of as simply specifying the address of the data, not as the actual physical source of the data that the address references.

6.5 Design for OPC Data Access Server: [8]

Figure 6.8 shows the OPC Server standard components as defined in the OPC Data Access specification. At its lowest level, the OPC Server is an I/O driver that can transmit data from the connected physical device (typically PLC). Another integral part of the OPC Server is a block optimizing the data collection while maintaining the highest possible performance of the PLC communication. Using the appropriate interfaces of the OPC Data Access specification, the transmitted data is then offered to the connected OPC Clients.

In some cases, OPC Servers are not designed for communication to specific devices. These OPC Servers do not include the two lower layers shown in the figure, which are, for example, substituted by a DDE protocol communication interface.

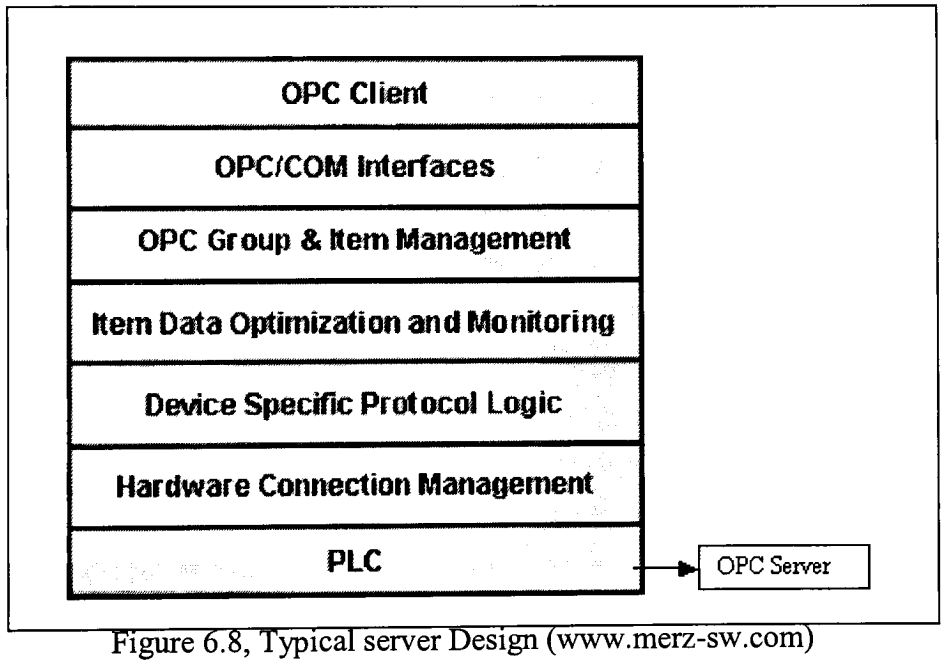


Figure 6.8, Typical server Design (www.merz-sw.com)

The server provides management of the OPC Groups and also controls, mediates and optimizes access to the physical devices by multiple clients. The Server is essentially an I/O Driver, which understands how to talk to some vendor specific data provider (hardware or software), and which exposes the data from this data provider via the standard OPC Interfaces.

6.6 Server Interfaces: [7]

OPC Data Access Specification provides specific interfaces for both the Server and Groups. Here we talk about the functionalities provided by the Server Interfaces. The methods of each interface have to be explored in order to get an idea of how the particular functionality is provided.

The Interfaces provided by the Server are:

- IOPCServer
- IOPCBrowseServerAddressSpace (optional)
- IOPCCommon
- IOPCItemProperties
- IconnectionPointContainer

The server interfaces listed above, allow the application to:

- Create and delete groups
- Browse the available tags
- Determine the attributes or fields associated with a tag
- Indicate the default local (National Language) to be used
- Translate error codes into meaningful text
- Obtain status information about the server
- Be notified when the server shuts down

6.6.1 IOPCServer

The methods provided by the IOPCServer interface are:

- AddGroup
- RemoveGroup
- GetGroupByName
- CreateGroupEnumerator
- GetErrorString
- GetStatus

The IOPCServer Interface allows the client to manage Groups. This is done using the Add, Remove, Get, and Enumerator functions. GetErrorString allows the client to translate errors (HRESULTS) into strings. GetStatus allows the client to monitor the overall status of the server.

6.6.2 IOPCBrowseServerAddressSpace

The methods provided by IOPCBrowseServerAddressSpace interface are:

- QueryOrganization
- ChangeBrowsePosition
- BrowseOPCItemIDs
- GetItemID

The browser interface (which is optional but which any worthwhile server should provide) allows the application to browse the tag names and attributes available within the server. It is carefully designed to work with essentially any organization; from a flat list of PLC registers to a hierarchical area/group/loop organization to a fully object oriented database. It does this by mapping all of the above into a hierarchical name space. This is intended to allow the application to present the results in an Explorer like browse tree. QureryOrganization tells the client if the space is flat or hierarchical. If the space is hierarchical, ChangeBrowsePosition moves up and down the hierarchy. BrowseOPCIDs returns an enumeration of the leafs or branches at a particular node. GetItemID returns the fully qualified name of an item. (This allows complete independence from the syntax and delimiters used by the vendor).

6.6.3 IOPCCommon

The methods provided by IOPCCommon interface are:

- Get/SetLocaleID
- QueryAvailableLocaleIDs
- GetErrorString
- SetClientName

The IOPCCommon interface will be common to all OPC servers. It allows control of Local ID for servers that support localization. It also allows you to translate errors (HRESULTS) into strings by calling GetErrorString (duplicate function to IOPCServer). Finally, it provides a diagnostic function, which allows the client to tell the server its name.

6.6.4 IOPCItemProperties

The methods provided by IOPCItemProperties interface are:

- QueryAvailableProperties
- GetItemProperties
- LookupItemProperties

Many systems associate a number of properties with an Item or Tag such as EGU limits, descriptors, etc. This interface allows the client to determine what properties are associated with each Item. It provides a convenience function, which allows those properties to be read. It also provides a way to determine if those properties have associated Item IDs of their own. For example, the Descriptor property of a tag might have an associated Item ID of "FIC101.DESC".

6.6.5 IConnectionPointContainer

The methods provided by IConnectionPointContainer interface are:

- EnumConnectionPoints
- FindConnectionPoint

The Connection Point Container on the server allows the client to indicate that he wants to be notified if the server shuts down in a controlled fashion. Connection Points are a standard Microsoft technique for establishing "callbacks" or event notifications to an object.

6.7 Group Interfaces: [7]

We take a look at the functionalities provided by the OPC Group COM object Interfaces. The methods of each interface have to be explored in order to get an idea of how the particular functionality is provided.

The interfaces contained by the OPC Groups are:

- IOPCGroupStateMgt
- IOPCItemMgt
- IOPCAsyncIO2
- IOPCItemMgt
- IOPCSyncIO
- IConnectionPointContainer

The Group interfaces allow the application to:

- Add and remove items from groups
- Manage the accuracy (update rate) of the data in the group
- Read or write values for one or more items in a group
- Subscribe to data in the group on an exception basis

6.7.1 IOPCGroupStateMgt

The methods provided by the IOPCGroupStateMgt interface are:

- GetState
- SetState
- SetName
- CloneGroup

After a client creates a group, IOPCGroupStateMgt allows control of the behavior of that group. It can change the name or update rate and a few other parameters of the group. It can also activate or deactivate the group. For example, an HMI might want to deactivate a group (but not delete it) if the display page using that group was minimized or hidden. CloneGroup can also create a copy of the group.

6.7.2 IOPCItemMgt

The methods provided by the IOPCItemMgt interface are:

- AddItems
- ValidateItems
- RemoveItems
- SetActiveState
- SetClientHandles
- SetDataTypes
- CreateEnumerator

IOPCItemMgt allows the client to add items to the group. For example, an HMI might create a group for each display page and then add to those groups all of the items referenced by that page. The most important functions here are AddItems and RemoveItems. The other functions are basically added features provided.

6.7.3 IOPCAsyncIO2

The methods provide by IOPCAsyncIO2 are:

- Read
- Write
- Refresh2
- Cancel2
- SetEnable
- GetEnable

Three communication mechanisms have been defined for OPC Data Access: asynchronous read/write, refresh and subscription. When a client issues an asynchronous read call, the server immediately returns control to the calling thread and later, using a callback mechanism, sends the requested values. The synchronous and asynchronous calls require that clients specify the list of process data that have to be read or written. Refresh and subscription are callback mechanisms. They are used to access predefined sets of process data.

Refresh is a "pull" mechanism; subscription is a "push" one. When an OPC client issues a refresh call, the OPC server asynchronously returns the current values of the predefined set of data points using a predefined communication path. Subscription is an event-based mechanism. OPC servers notify clients when significant changes (i.e., bigger than the dead-band defined for the OPC group) occur within a predefined set of process data.

Cancel can cancel an operation in progress if for example the client wants to shut down. Set/Get Enable allows the client to turn the subscription on or off. This allows the client to perform asynchronous I/O independently of the subscription i.e., to turn off OnDataChange without turning off Read/Write complete.

6.7.4 IOPCSyncIO

The methods provided by IOPCSyncIO interface are:

- Read
- Write

When a client issues a synchronous read call, the server does not return the control to the calling thread until sending back the requested values. This function runs to completion. The user provides a list of items and values to write and get back as list of errors.

6.7.5 IConnectionPointContainer

The methods provided by IConnectionPointContainer interface are:

- Advise
- Unadvise

The Connection Point Container on the group allows the client to indicate that it wants to be notified on changes to data or on the completion of asynchronous read or write operations. OPC requires that vendors provide the Advise and Unadvise methods. These are used to hook up callbacks, which are used by the asynchronous and subscription modes of data access. Clients, which perform synchronous reads, do not need to use this

interface. One should note that the groups operate independently so on could poll, subscribe or do asynchronous I/O to different groups.

Connection Points are a standard Microsoft technique for establishing "callbacks" or event notifications to an object.

6.8 Client side Interfaces: [7]

The OPC Data Access Specification also specifies two COM interfaces that the server could call towards the client side.

The interfaces contained by the Client side are:

- IOPCShutdown
- IOPCDataCallback

6.8.1 IOPCShutdown

The method contained by the IOPCShutdown is:

- ShutdownRequest

This interface allows the server to tell the client that an orderly shutdown has been requested. The server can provide a text string indicating the reason for the shutdown. The client can delete all groups, items and subscriptions (release all interfaces) to allow a clean shutdown.

6.8.2 IOPCDataCallback

The methods contained by the IOPCDataCallback are:

- OnReadComplete
- OnWriteComplete
- OnCancelComplete
- OnDataChange

On the client side, there is one callback interface required if one performs asynchronous or exception based I/O. OnReadComplete is called when Asyncio2 Read operation completes. OnWriteComplete is called when Asyncio2 Write operation completes. OnCancelComplete is called when Asyncio2 Cancel2 operation completes for a previously started read, write or refresh. OnDataChange is called whenever one or more values change in a group (subject to deadband and update rate).

References:

1. Santori, M. – "OPC: OLE for Process Control", Real Time Magazine, April 1997 <http://www.realtime-info.be/magazine/97q4/1997q4_p078.pdf>
2. Component Object Model – TechTarget, 2002 <http://searchwin2000.techtarget.com/sDefinition/0,sid1_gci211823,00.html>
3. R. Barillère; V. Baggiolini; M. Beharell; D. Chmielewski; P. Gras; H. Milcent; K. Kostro. – "Results of the OPC Evaluation done within JCOP for the Control of the LHC Experiments" International Conference on Accelerator and Large Experimental Physics Control Systems, 1999, Trieste, Italy <http://www.elettra.trieste.it/sites/icalepcs99/proceedings/papers/mc1p43.pdf>
4. A Technical Overview of the OPC Data Access Interfaces – Intellution Inc, 1998 <http://www.intellution.com/opchub/opc_interfaces_overview.asp>
5. OPC/OLE for Process Control Overview – Intellution Inc, 1998 <http://www.intellution.com/opchub/opc_wbfoverview.asp>
6. OPC Task Force. (1998). "OPC Overview, Version 1.0". OPC Foundation < http://www.iconics.com/support/pdfs/opc_specs/opcda20.pdf>
7. OPC Data Access 2.0 Technical Overview Presentation – Intellution Inc, 1998 <http://www.intellution.com/opchub/opc_data_access_presentation.asp>
8. Merz – OPC Server According to Data Access 1.0a 2.0 Specification – Merz, 1999. <http://www.merz-sw.com/articles/opc_da.php3>

7. OPC Case Studies

In this chapter, we discuss some applications (case – studies) to illustrate the potential of OPC based systems integration. The following four case studies are presented:

1. Remote access to a group of electric motors from a wireless phone
2. SAP OPC Data Access linking process related data with business information systems
3. Combining OPC with Autonomous Decentralized Systems
4. Integration of Fieldbus Systems in Computer-Aided Facilities Management

7.1 OPC – Based WAP Solution:

This case study discusses the application of OPC – based remote automation systems using advanced telecommunications technique like WAP (Wireless Application Protocol). The WAP protocol is the leading standard that enables users to share web – based information and services on wireless terminals like digital mobile phones.[1]

7.1.1 WAP operated OPC – based Automation System Application: [2]

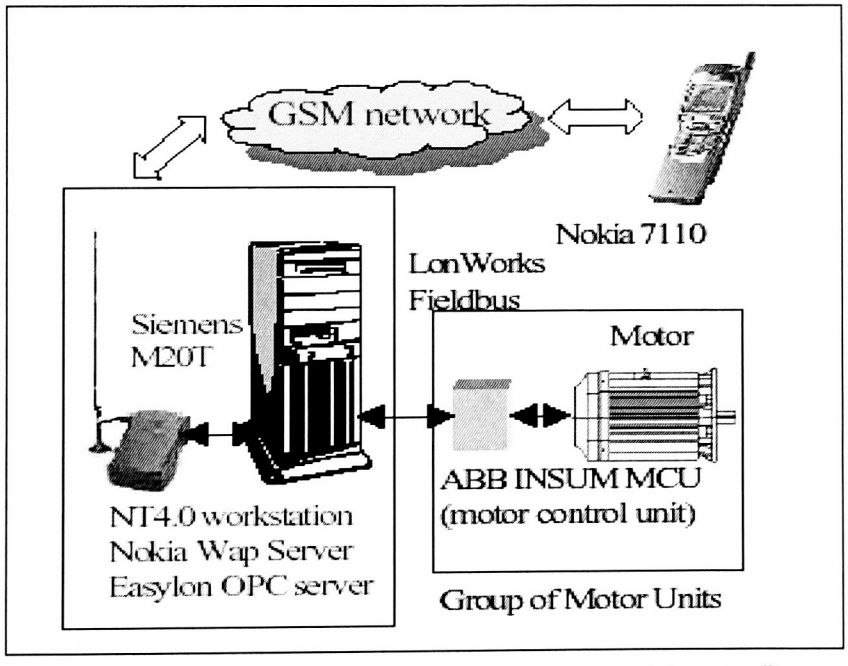


Figure 7.1, System Architecture (www.telecomlab.oulu.fi)

In this application a group of motors are accessed and controlled from a wireless phone device using WAP protocol. As shown in figure 7.1, the application consists of a Nokia 7110 WAP phone, a Windows NT 4.0 workstation running a Nokia WAP Server, a Siemens M20T GSM modem to be used as a Remote Access Service (RAS) device to answer incoming data calls and to authenticate a WAP phone user, Easylon OPC Server 2.05, LonWorks fieldbus that connects from the PC to a group of electric motors through ABB Insum motor control unit[2]. The Easylon OPC Server enables standardized access to the LonWorks control network thus allowing an OLE – compatible program to visualize, control and monitor the process (in this case a group of motors) connected to a LonWorks control network.[3] Java was selected in the design phase taking advantage of its platform independency.

7.1.2 WAP: [2]

The basic idea of WAP is to make available services provided by the Internet to mobile users. It defines the application environment, the communication protocol, as well as an application layer consisting of a language called the Wireless Markup Language (WML) and scripting language (WMLS). WAP gives due attention to the limitations of mobile phones like small displays, slow data transmission and limited input capabilities. The key elements of the network structure involving WAP protocol are shown in figure 7.2.

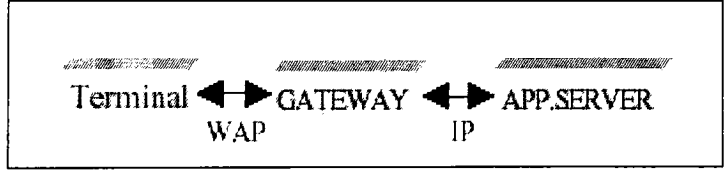


Figure 7.2, Network Structure (www.telecomlab.oulu.fi)

The key element in the WAP environment is the WAP gateway, which primarily controls the security of the network, routing the request from the terminal device to the application server and back, and to convert data frames obtained from a protocol to a format that could be understood by another protocol. Here the gateway converts the data packets from WAP protocol to TCP/IP protocol, thus translating WAP requests to HTTP requests.

A simple WAP application would consist of a client application (WML and WMLS pages) that is downloaded by the WAP gateway from the application server and passed to the user agent (mobile device) for execution. Application are made more efficient with server side enhancements like CGI scripts, servlets etc. In this application, WAP is used in remote automation to control and monitor devices connected to the Internet. In applications where the user controls the process, it is important that security is guaranteed so that no one else can control the process than an authorized user.

7.1.3 Implementation: [2]

In this application, Java servlets are implemented to generate WML-based user interface so as to create as many instances of Java OPC Client objects as there are motors specified to be on the LonWorks network. The implementation of these Java OPC clients is based on Java2opc software, which is the bridge that allows developers to write java application to any OPC Server. The software that was implemented is running inside the Nokia WAP server servlet engine. Figure 7.3, shows the basic idea of the software architecture.

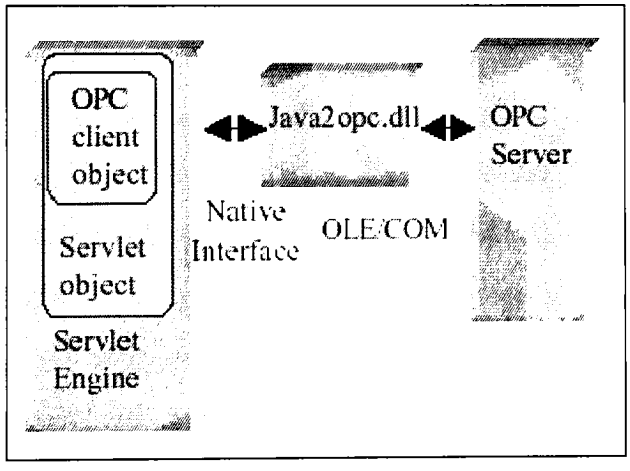


Figure 7.3, Software Architecture (www.telecomlab.oulu.fi)

Using the above software architecture, a WAP phone user can select the motor that is to be controlled and could then either start, stop or reset the motor. Thus the suitability of OPC to remote automation has been tested by this application.

7.2 SAP OPC Data Access:

This case study discusses closing the gap between the business systems for supply chain management and shop floor control systems, where enterprise-wide integration systems (ERP) can attain real time data from the process manufacturing area using OPC components in order to meet the daily business needs of the company.

7.2.1 What is Enterprise Resource Planning (ERP)?

Enterprise resource planning software, or ERP is software that attempts to integrate all departments and functions within a company into a single computer system so that it could serve all the different departments' needs and allows smoothening the interactions between various departments in an enterprise.

ERP is ideally a software program that serves the needs of people for example in the finance department as well as it does for the people in human resources and in the warehouse departments. Each of these departments typically has its own computer system optimized for the particular way that the department does its work. But ERP combines them all together into a single, integrated software program that runs off a single database so that the various departments can more easily share information and communicate with each other.[4]

SAP AG is the world's largest solution provider that develops e-business or ERP software's like mySAP.com and SAP R/3 that focuses on integrating various processes involved within an enterprise. It also works towards improving various collaborative processes between enterprises. [5]

7.2.2 The SAP ODA Application: [6]

SAP ODA takes advantage of the flexibility and agility provide by the Internet for manufacturing, in order to become a highly responsive part of the supply chain. They provide a link between a wide variety of systems on the shop floor (e.g., process controls, DCS, SCADA-systems, etc.) and the upper level business systems that forms the business management layer of the Process Control Information Architecture.

SAP OPC Data Access (referred as SAP ODA) is a piece of software that is called from a mySAP.com component that delivers a gateway to the OPC-world and enables its solution for the OPC-technology. Its primary goal is to tighten the dynamic link between business systems for supply chain management and shop floor systems. With the SAP ODA software no additional customer programming would be needed to address the wide variety of available OPC-servers on the market.

The SAP ODA is an RFC-server for the SAP system and an OPC-client for the OPC-server. SAP customers who wish to write other applications that communicate with R/3 applications and databases can use the RFC (Remote Function Call) interface to do so.

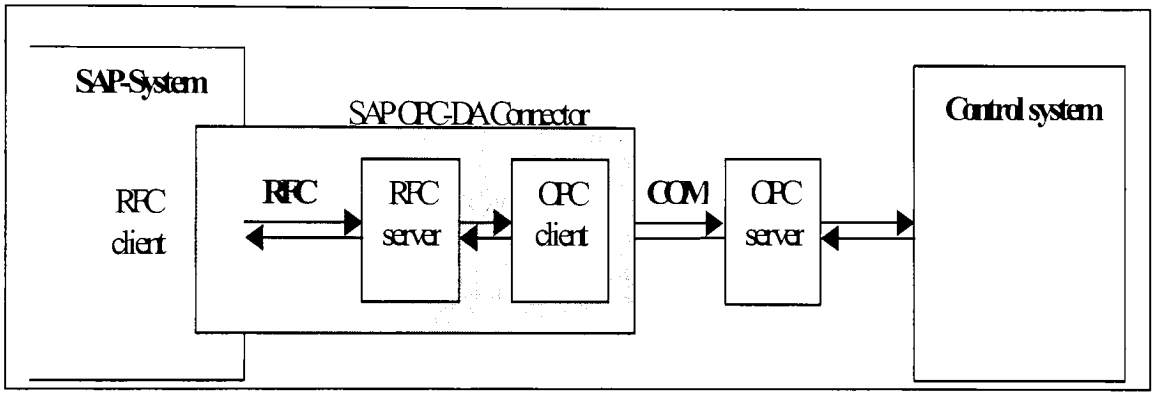


Figure 7.4, SAP ODA Interface (www.intellution.com)

The SAP ODA is especially designed for interactive synchronous processes. It does not replace SAP's existing interface to Process Control Systems (PI-PCS), which supports fully automated processes with a message-based communication. With the link to OLE for Process Control servers, SAP supports an additional industry standard. Both interfaces coexist for different purposes.

The SAP ODA provides four basic methods:

- Browse available local OPC-servers
- Browse for variable names (OPC-specification: namespace browsing) of a given OPC-server

- Read actual values of OPC items (please see OPC specification for details) from an OPC-server
- Write actual values of OPC-items to a given OPC-server

The SAP ODA-Connector is called from the mySAP.com component via SAP's Remote Function Call protocol. Methods are carried out in the SAP ODA where they are converted to the related OPC-interface calls. Data are transferred from/to the OPC-server.

SAP's browser-based PI Sheet is especially designed to support manual and semiautomatic processes in the process manufacturing area. From the PI Sheet for example a process operator can press a button to read or write values from/to any shop floor automation system supporting the OPC DA 1.0/2.x-specification.

No specific control system is cited in the application, but any control system with an OPC server interface could be used here.

7.3 Combining OPC with Autonomous Decentralized Systems: [7]

This case study discusses Autonomous Decentralized Systems (ADS) that has been successfully used in many distributed computing areas such as factory automation, traffic control, office automation, and nuclear power plants. Since ADS is a lower layer of control system as in the field management and process management layer in the Process Control Information Architecture, it falls short of interfacing with the business management needs. This case study presents a framework for combining OPC with ADS into the same computer in order to cater or make available process data for business management systems.

7.3.1 Autonomous Decentralized System (ADS): [7]

ADS aims for seamless integration of various software subsystems in the process and field management layers in process control applications. It achieves co-ordination with other software subsystems by communicating through a Data Field (DF) that circulates the data so that various software subsystems could select the data according to the content

code. A software subsystem in ADS is referred to as an Atom, where every Atom is connected to the data field. The Atom consists of management system software called Autonomous Control Processor (ACP). The ACP software architecture is shown in figure 7.5.

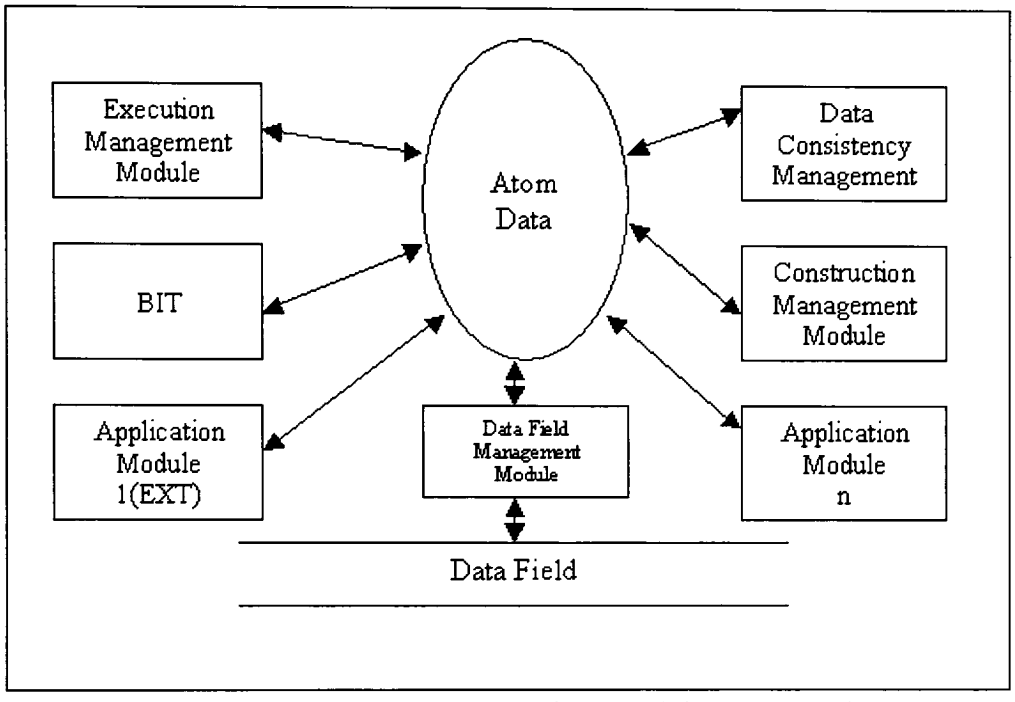


Figure 7.5, ACP Software Architecture (Sitao, W, et. al.)

Each ACP operates with its local information and communicates with other ACP's asynchronously by broadcasting messages in the data field. It consists of various modules that aid in effective data handling, program execution, on-line maintenance, on-line expandability and fault tolerance. Since ADS is in charge, mainly of the lower layers of the computer control system, a standard interface is required for passing data from shop-floor devices and machines up to business information and control systems. OPC would play a decisive role in forging the connectivity required to the higher layers of the computer control system.

7.3.2 Applying OPC over ADS Application: [7]

OPC is used at the upper layers such as the business management and process management layers through COM/DCOM technology. OPC clients resident at the upper layer monitor, control, administer, expand and delete necessary devices at lower layers using the OPC server. ADS, here is the basis for the lower layer of the system. OPC server acts as a gateway between upper layer and lower layer of the control system. The OPC server architecture by combining OPC with ADS is shown in figure 7.6.

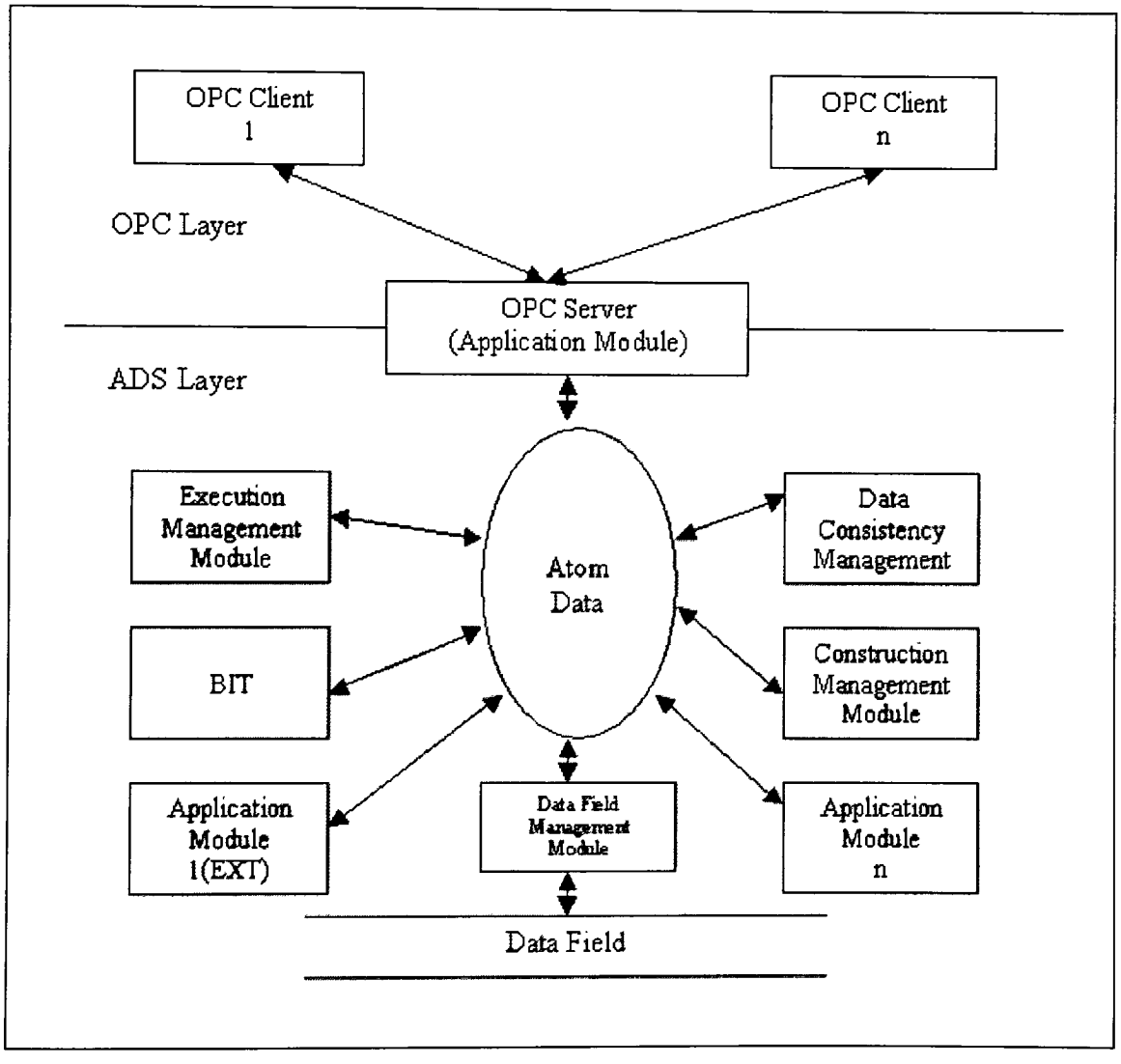


Figure 7.6, Applying OPC over ADS (Sitao, W, et. al.)

At the lower level the OPC server operates as an application module with ADS, communicating data through message broadcast with all the online devices in the system.

Thus by combining OPC with ADS, a new software architecture has been created that has advantages from two technologies, having the interoperability, efficiency and speed provided by OPC and on-line maintainability, expandability and fault tolerance provided by ADS.

7.4 Integration of Fieldbus Systems in Computer-Aided Facilities Management: [8]

This case study shows an approach for the integration of different fieldbus systems into the technical Facility Management system. This integration is realized using standardized mechanisms like OPC technology and common object definitions for different fieldbus systems. Computer-Aided Network Facilities Management (CANFM) is a system used for planning, documenting and managing of data and telecommunication networks. The paper shows the possible advantages of integrate or combining the very complex CANFM systems with the fieldbus systems.

7.4.1 CANFM: [8]

As mentioned before, CANFM is used for planning, documenting, and managing data and telecommunication networks. It manages all active and passive components in a network (e.g. computer, peripheral devices, telecommunication equipment, switch, routers, hubs etc.) as well as the connections between them. Fieldbus systems are currently not considered as an essential network facility, since they are considered as a feature of automation systems where a closed solution (focused only on automation systems) has been established. Nowadays, more and more field devices are equipped with fieldbus interfaces, making an increasing need to consider the integration of CANFM systems with fieldbus systems when preparing new systems for a comprehensive facility management.

Integration of Fieldbus systems into CANFM systems would mean that:

- Management information of the field devices as device information (vendor, revision number etc) and status information (device state, status of a field value etc) has to be provided to the Facility Management for the purpose as defined by such system

- Maintenance of fieldbus systems and its components can be supported based on the management of the position of field devices, tags assigned to transmitters and actuators, device types and their device management information and the installation information for connections between the devices
- The topology and the installation layouts can be documented in an automated way.

To reach the above-mentioned advantages, one would need to combine the complex CAMFM systems with fieldbus systems. OPC technology is thereby used here to define an interface in order to exchange data between the two systems.

7.4.2 Approach for Integration (Application): [8]

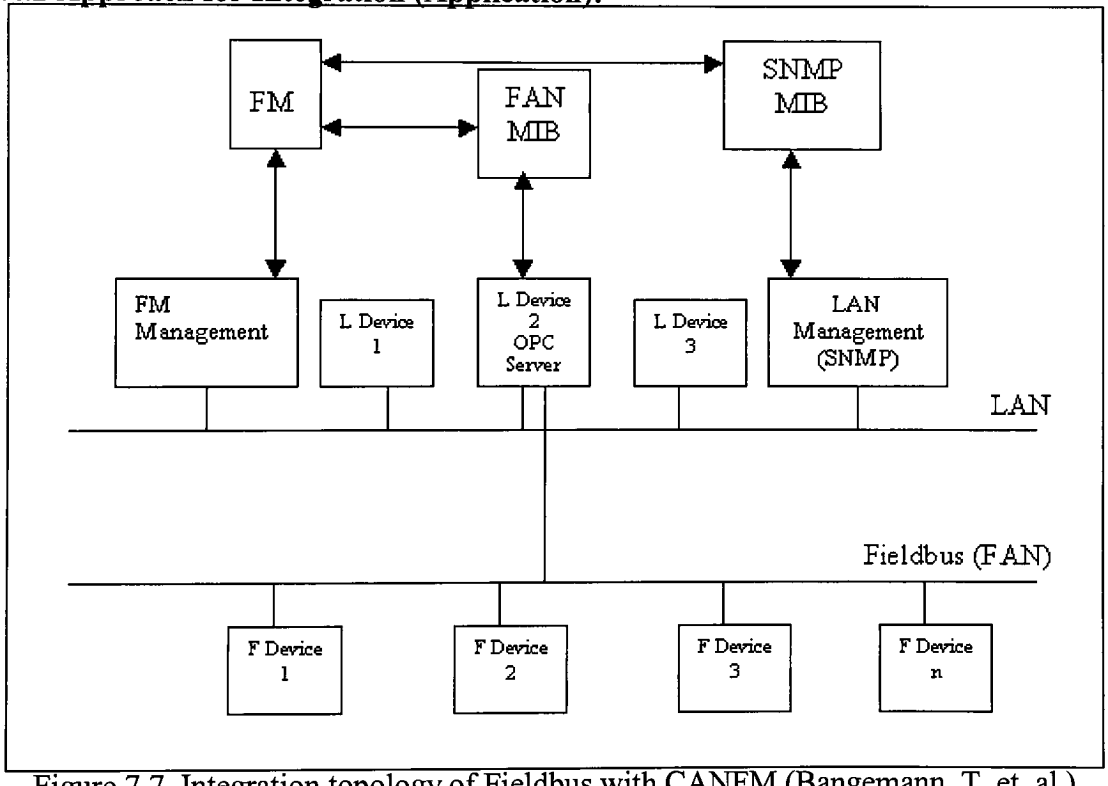


Figure 7.7, Integration topology of Fieldbus with CANFM (Bangemann, T, et. al.)

For Facility Management, management information is basically the information that can be used by the CANFM system to manage fieldbus equipments in a facility. Acquiring

the information based on the OPC mechanism makes it possible to gain dynamic data from the facilities on-line.

The managed objects or network components are ideally selected based on the requirements of Facility Managers. Managed objects can be monitored, controlled and has the capability of reporting events. The parameters obtained from the managed objects are collected within a Management Information Base (MIB). The information residing in the Management Information Base can be obtained and/or modified via a network management protocol (in this case SNMP – Simple Network Management Protocol).
The managed objects could be object types like Application Process Objects, Communication Stack Objects, and Resource Objects. Application Process Objects have all the parameters to describe the application, Communication Stack Objects consists of parameters for communication between applications that want to exchange data, and Resource Objects describe the device and the available resources.

The instances of managed objects are distributed in the complete fieldbus system. In order to achieve unique access to objects from different fieldbus systems, the OPC server performs the task of decoupling both the fieldbus and CANFM systems. The server has a universal interface that is visible to all the clients and a fieldbus specific interface for each fieldbus system. Thus the OPC server would provide an image of all the management information for each fieldbus system as requested by the OPC client located in the CANFMS.

Thus the application shows us the ability of OPC technology in integrating different fieldbus systems to the Computer-Aided Network Facility Management System.

References:

1. WAP/WML Tutorial, "WAP Tutorial" - Refsnes Data, 1999 – 2002 < http://www.w3schools.com/wap/default.asp>
2. Kero, J.; Koivisto, H.; Kulmala, H. – "OPC-Based WAP Solution", Tampere University of Technology Automation and Control Institute < http://www.telecomlab.oulu.fi/~ojalam/fwcwpapers/p20.pdf>
3. Easylon OPC Server - Gesytec GmbH Aachen, Germany, 2001 <http://www.gesytec.com/englisch/index-e.htm>
4. Koch, C – "The ABCs of ERP" Enterprise Resource Planning Center, CXO Media Inc., 2002 <http://www.cio.com/research/erp/edit/erpbasics.html>
5. SAP AG Corporate Review – SAP AG, 2002 <http://www.sap.com/company/profile_long.htm>
6. SAP OPC Data Access (SAP ODA) 1.0 – SAP AG, 2000 < http://www.intellution.com:84/downloads/special/features/sapoda_paper.doc>
7. Sitao, W.; Qingquan, Q.; – "Combining OPC with autonomous decentralized systems" International Workshop on Autonomous Decentralized Systems, 2000. Pages: 126 – 129
8. Bangemann, T.; Hahniche, J.; Neumann, P. – "Integration of fieldbus systems in computer-aided facility management" Industrial Electronics Society, 1998. IECON '98. Proceedings of the 24th Annual Conference of the IEEE, Volume: 3, 1998. Pages: 1835 -1840

8. OPC Application

In this application, we demonstrate the ability of OPC technology in providing smooth integration of two different systems, and thereby facilitating access to data originating from a particular system by another system. Interoperability of devices can be realized if and only if there exists OPC compatible interfaces between both the systems (the data provider and the data consumer).

8.1 Application Description:

We demonstrate the integration of Siemens Simatic S7 PLC (Programmable Logic Controller) with LabVIEW 6i, which is a graphical programming development environment based on the G programming language for data acquisition and control, data analysis, and data presentation. LabVIEW is a powerful graphical programming language that provides a way to develop control applications without much difficulty and complexity that are otherwise inherent with text based programming languages, thus making the programming comparatively intuitive to scientists and engineers. LabVIEW would be used here as a client application that makes use of the OPC tags created in the Siemens environment, to be tied to various objects created in the Front Panel of a LabVIEW application in order to act as a good and efficient Human Machine Interface to the process being controlled by the Siemens Soft PLC.

The process is controlled by the PLC Ladder logic that resides in the Siemens Step7 Soft PLC. It is called a Soft PLC, since the main processor of the PLC resides in one of the bays or slots of a computer in the form of a PCI Card (CP-5613). The conventional stand-alone PLC has the processor, memory and I/O modules all residing in a single chassis. The PCI card of the soft PLC is connected to the various I/O modules (the I/O modules are tied to the process directly) through a PROFIBUS cable and an interface module (IM153-1).

The LabVIEW environment is used here to act as an interface to the process controlled by the Siemens Soft PLC, making use of its superior graphical interfacing capabilities.

The DataSocket capability of LabVIEW is utilized to subscribe to tags generated by the OPC Server residing in the Siemens WinAC operating environment.

8.2 Application Architecture:

In this application, we see how two different systems (LabVIEW and Siemens Step7 PLC) are integrated in order to take advantage of the capabilities provided by them individually.

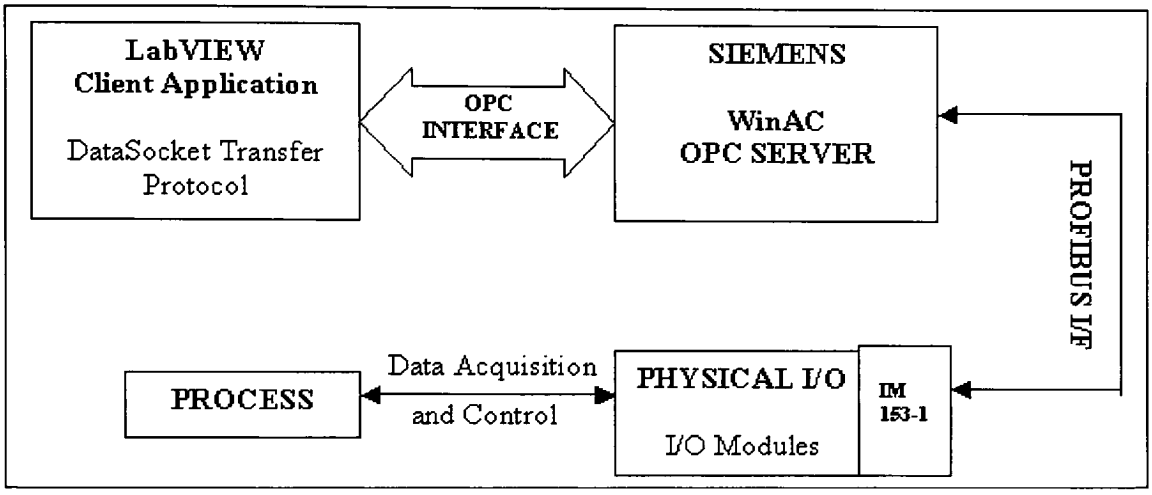


Figure 8.1, Application Architecture

The schematic of the application architecture is shown in Figure 8.1. The DataSocket Transfer Protocol provided in LabVIEW is utilized by the client application to subscribe from or publish to any OPC items or tags residing in the OPC server, hence one would not only be able to acquire data from the process but also would be able to control the process. Here, the OPC Server is provided by WinAC (Windows Automation Center), which is part of the Siemens Step7 suite. WinAC is the operating system environment that is utilized to gather data from the process. As seen in figure 8.1, a PROFIBUS cable connects the soft PLC PCI card with the Interface Module (IM153-1) providing the interface between the soft-PLC and the I/O Modules in the field, which in turn is connected to the process.

8.3 Application Software Requirements:

Tags (data items) are created by making use of the Symbol table functionality provided by the Step7 software within the WinAC environment. Symbolic tables contain symbolic names of the I/O addresses associated with the I/O modules that are connected to the various sensors or actuators in the process. An example of a symbolic table is shown in Figure 8.2.

	Symbol	Address	Data type	Comment
1	Output 1	Q 0.0	BOOL	Output
2	Output 10	Q 1.1	BOOL	Output
3	Output 11	Q 1.2	BOOL	Output
4	Output 12	Q 1.3	BOOL	Output
5	Output 13	Q 1.4	BOOL	Output
6	Output 14	Q 1.5	BOOL	Output
7	Output 15	Q 1.6	BOOL	Output
8	Output 16	Q 1.7	BOOL	Output
9	Output 2	Q 0.1	BOOL	Output
10	Output 3	Q 0.2	BOOL	Output
11	Output 4	Q 0.3	BOOL	Output
12	Output 5	Q 0.4	BOOL	Output
13	Output 6	Q 0.5	BOOL	Output
14	Output 7	Q 0.6	BOOL	Output
15	Output 8	Q 0.7	BOOL	Output
16	Output 9	Q 1.0	BOOL	Output
17	Sim Input 1	I 0.0	BOOL	Input from Simulator
18	Sim Input 10	I 1.1	BOOL	Input from Simulator
19	Sim Input 11	I 1.2	BOOL	Input from Simulator
20	Sim Input 12	I 1.3	BOOL	Input from Simulator
21	Sim Input 13	I 1.4	BOOL	Input from Simulator
22	Sim Input 14	I 1.5	BOOL	Input from Simulator
23	Sim Input 15	I 1.6	BOOL	Input from Simulator
24	Sim Input 16	I 1.7	BOOL	Input from Simulator
25	Sim Input 2	I 0.1	BOOL	Input from Simulator
26	Sim Input 3	I 0.2	BOOL	Input from Simulator
27	Sim Input 4	I 0.3	BOOL	Input from Simulator
28	Sim Input 5	I 0.4	BOOL	Input from Simulator
29	Sim Input 6	I 0.5	BOOL	Input from Simulator
30	Sim Input 7	I 0.6	BOOL	Input from Simulator
31	Sim Input 8	I 0.7	BOOL	Input from Simulator
32	Sim Input 9	I 1.0	BOOL	Input from Simulator

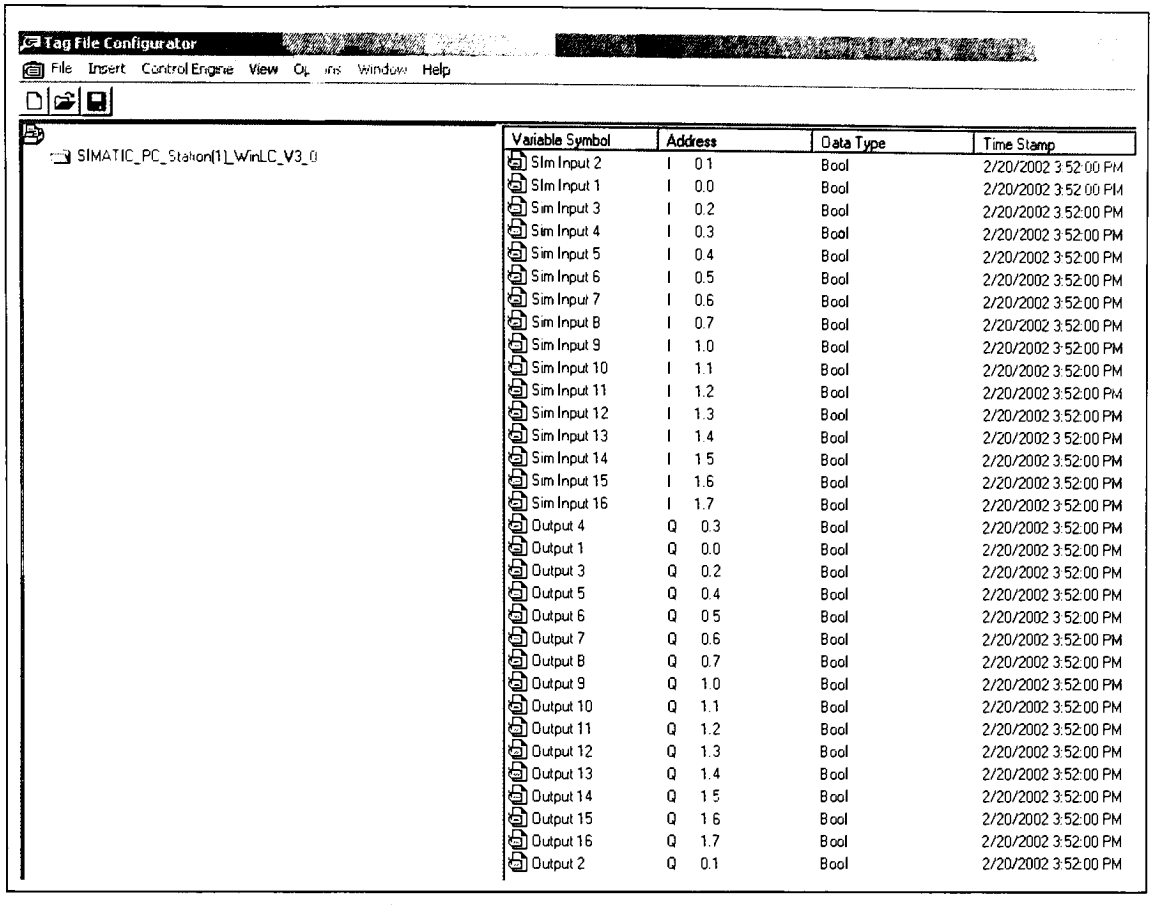
Figure 8.2, Symbols table in Step7

Figure 8.2 shows the symbolic names given to the various I/O points or addresses in the I/O modules (here Q represents Output point, while I represents an Input point). For

example symbolic name "Output 1" corresponds to the output point residing on output module 0 and point 0. The data type shown here is type Boolean.

Symbolic names are given to I/O points in order to create tags on the WinAC OPC Server, so that other applications could access the tags, be it on the same computer or residing on another computer in the same network. Tags are central elements for applications to access process values. A tag is an item, which is assigned a logical connection from the process I/O. This connection determines which channel delivers the process values to the tags using which connection. Thus tags are used to acquire external process values and deliver it to client applications via the OPC Interface.

After the symbols are created for the I/O points connected to the process, the next logical step would be to convert the symbolic name to OPC tags. This is done with the help of the Tag File Configurator software provided by the Siemens Simatic Software Suite. The Tag File Cofigurator software is used to browse to the Step7 PLC program that contains the symbolic names and then convert the I/O symbolic names to OPC tags. The conversion of the symbols in the symbolic table shown in Figure 8.2 is shown in Figure 8.3.



Variable Symbol	Address	Data Type	Time Stamp
Sim Input 2	I 0.1	Bool	2/20/2002 3:52:00 PM
Sim Input 1	I 0.0	Bool	2/20/2002 3:52:00 PM
Sim Input 3	I 0.2	Bool	2/20/2002 3:52:00 PM
Sim Input 4	I 0.3	Bool	2/20/2002 3:52:00 PM
Sim Input 5	I 0.4	Bool	2/20/2002 3:52:00 PM
Sim Input 6	I 0.5	Bool	2/20/2002 3:52:00 PM
Sim Input 7	I 0.6	Bool	2/20/2002 3:52:00 PM
Sim Input 8	I 0.7	Bool	2/20/2002 3:52:00 PM
Sim Input 9	I 1.0	Bool	2/20/2002 3:52:00 PM
Sim Input 10	I 1.1	Bool	2/20/2002 3:52:00 PM
Sim Input 11	I 1.2	Bool	2/20/2002 3:52:00 PM
Sim Input 12	I 1.3	Bool	2/20/2002 3:52:00 PM
Sim Input 13	I 1.4	Bool	2/20/2002 3:52:00 PM
Sim Input 14	I 1.5	Bool	2/20/2002 3:52:00 PM
Sim Input 15	I 1.6	Bool	2/20/2002 3:52:00 PM
Sim Input 16	I 1.7	Bool	2/20/2002 3:52:00 PM
Output 4	Q 0.3	Bool	2/20/2002 3:52:00 PM
Output 1	Q 0.0	Bool	2/20/2002 3:52:00 PM
Output 3	Q 0.2	Bool	2/20/2002 3:52:00 PM
Output 5	Q 0.4	Bool	2/20/2002 3:52:00 PM
Output 6	Q 0.5	Bool	2/20/2002 3:52:00 PM
Output 7	Q 0.6	Bool	2/20/2002 3:52:00 PM
Output 8	Q 0.7	Bool	2/20/2002 3:52:00 PM
Output 9	Q 1.0	Bool	2/20/2002 3:52:00 PM
Output 10	Q 1.1	Bool	2/20/2002 3:52:00 PM
Output 11	Q 1.2	Bool	2/20/2002 3:52:00 PM
Output 12	Q 1.3	Bool	2/20/2002 3:52:00 PM
Output 13	Q 1.4	Bool	2/20/2002 3:52:00 PM
Output 14	Q 1.5	Bool	2/20/2002 3:52:00 PM
Output 15	Q 1.6	Bool	2/20/2002 3:52:00 PM
Output 16	Q 1.7	Bool	2/20/2002 3:52:00 PM
Output 2	Q 0.1	Bool	2/20/2002 3:52:00 PM

Figure 8.3, Tag File Configurator

Figure 8.3 shows the OPC tags created after the Tag File Configurator converted the symbolic names. This file is then saved in the form of a ". tsd" file, which is later called or made available to other applications in the network with the help of the Computing Configuration software provided by the Step7 Software Suite. The Computing Configuration software browses for the ". tsd" file that was created using Tag File Configurator software, in order to create a connection via a Tag Source File. This is depicted in Figure 8.4.

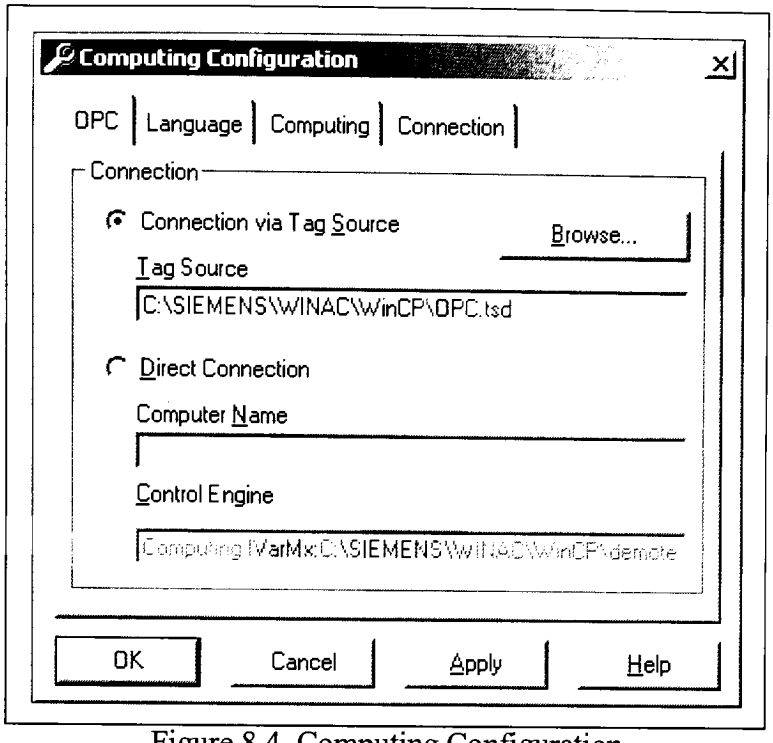


Figure 8.4, Computing Configuration

After the tags are made the available in the network, now the development of the interface is done on LabVIEW. LabVIEW provides a very superior front panel graphical interfacing capability. Front panel objects can be selected from the Control palette and dropped on the Front Panel, to create an attractive Human Machine Interface that allows an operator to view the process from his/her control room. The networking capability or the web publishing capability of LabVIEW can also be utilized to create dynamic web pages of the interface in order to make the process to be visible to managers present at a remote location. Thus with LabVIEW one could collect data at a location, analyze it in a separate location and present it in a different location. The Front Panel of the application is shown in Figure 8.5.

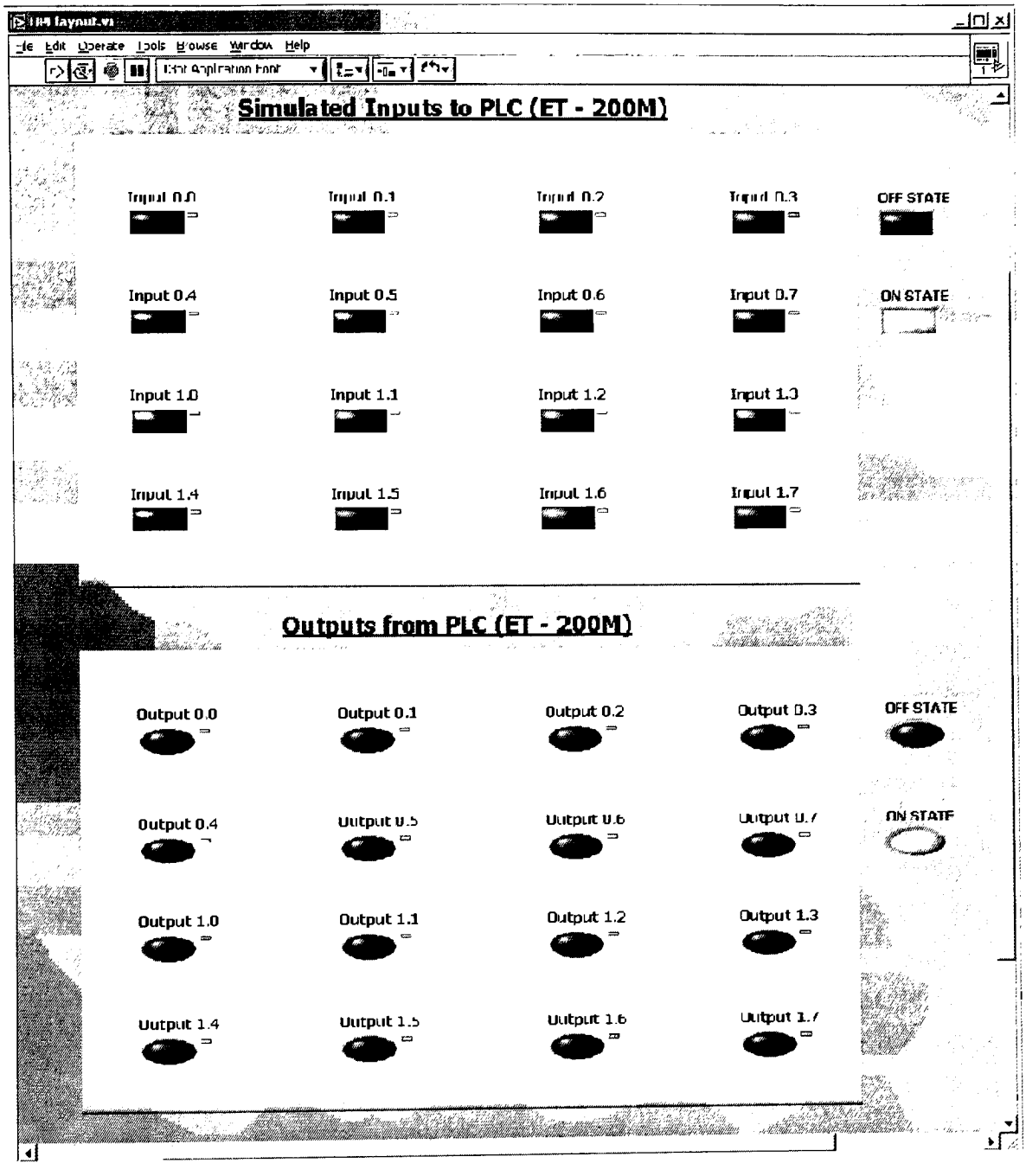


Figure 8.5, Front Panel to the process

Each indicator shown in the Front Panel (Figure 8.5) is connected to the OPC tag file source data item residing in the Siemens environment by way of the DataSocket Transfer functionality provided in LabVIEW. Here more than one object can subscribe to a single OPC data item or tag, or a single LabVIEW interface can access OPC tags residing in

different ".tsd" files or different OPC Servers in the Siemens environment. The connection to the OPC data items residing in the WinAC OPC Server, from the LabVIEW client application is accomplished by subscribing from or publishing to the OPC server via the datasocket connection functionality provided by the LabVIEW DataSocket Transfer Protocol. The DataSocket Connection dialog box that appears when one right clicks on the front panel object in order to establish a connection with the OPC data item is shown in Figure 8.6.

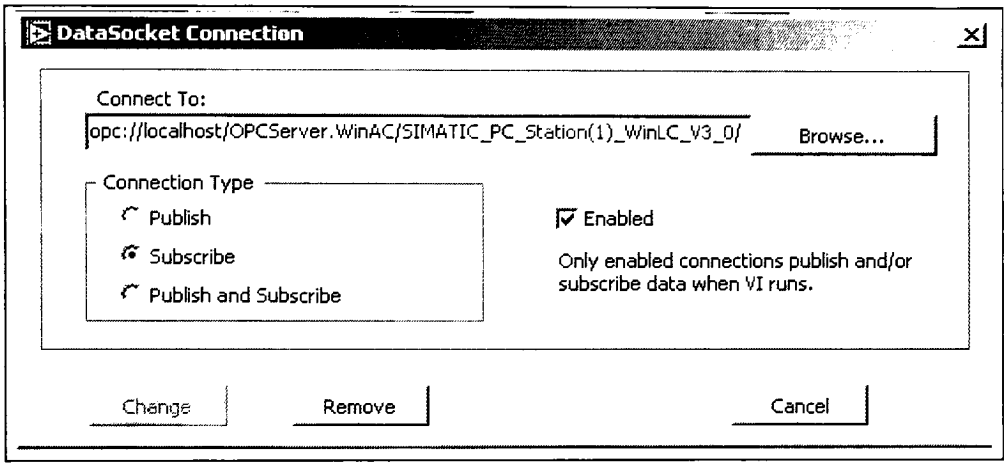


Figure 8.6, DataSocket Connection in LabVIEW

With the "Browse..." button one could browse to the OPC data item resident on a server located either locally (as shown in Figure 8.6) or remotely in the same network. The path shows that the data item is subscribed from a WinAC OPC Server residing on the local computer.

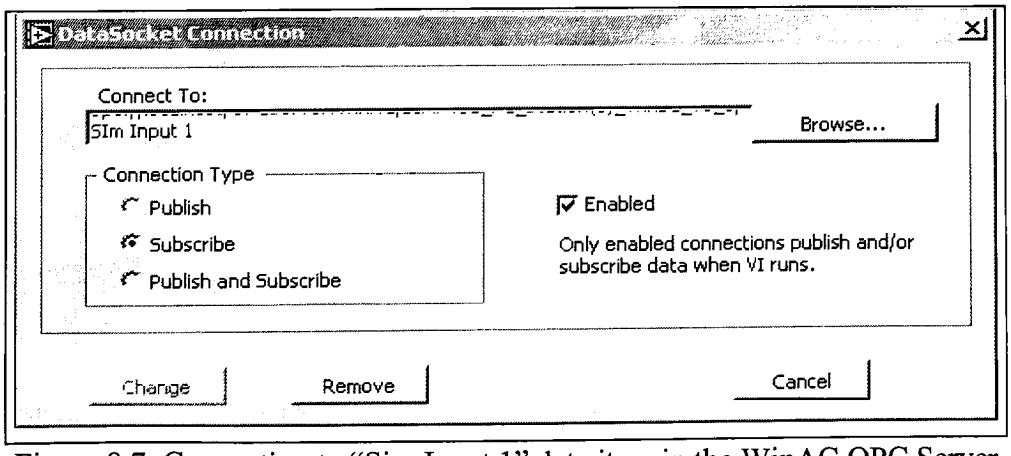


Figure 8.7, Connection to "Sim Input 1" data item in the WinAC OPC Server

The path to the data item in the OPC Server shows that the Front panel indicator is connected to tag named "Sim Input 1" residing in the WinAC OPC Server. This tag is connected to module 0 and input point 0, as we can see from the symbolic table (Figure 8.2).

After the two systems have been integrated by the OPC connection, any change in state in the process controlled by Siemens, would be seen in LabVIEW and made available to operators in the control room and managers in the business management level for presentation by way of the networking capabilities of LabVIEW.

With the above application we would be using the superior front-end graphical programming development environment, and excellent presentation and analysis capabilities provided by LabVIEW and at the same time would be making use of the robustness and stability provided by the Siemens backend PLC, proving the point that choosing the best possible framework for a customers manufacturing system is not a possibility but a reality with OPC technology.

8.4 Conveyor Experiment Application: [1]

Here we introduce a physical simulator (nicknamed "Conveyor Experiment") that simulates a single server queuing process. The physical simulator model is present in the Advanced Systems Integration Laboratory at Rochester Institute of Technology. Students use this experiment as part of the curriculum for their "Computer Integrated Manufacturing" course.

The conveyor experiment is controlled by the Step7 ladder logic that is provided in the Siemens Simatic S7 programming suite. The inputs from the process, the outputs to the process and the counter values are utilized by the WinAC OPC Server to generate OPC tags or data items in order to provide a data channel for other applications. LabVIEW data acquisition software would utilize these data items or tags in order to represent the process being monitored and controlled on the human machine interface developed in it.

8.4.1 Experimental Setup:

The Experimental setup of the conveyor experiment consists of two conveyors that are moved by motors in the same direction. It also has three arms (loading arm, transfer arm, and sorting arm) that can run either in the forward or backward direction with the help of motors. The schematic layout of the experiment is shown in Figure 8.8.

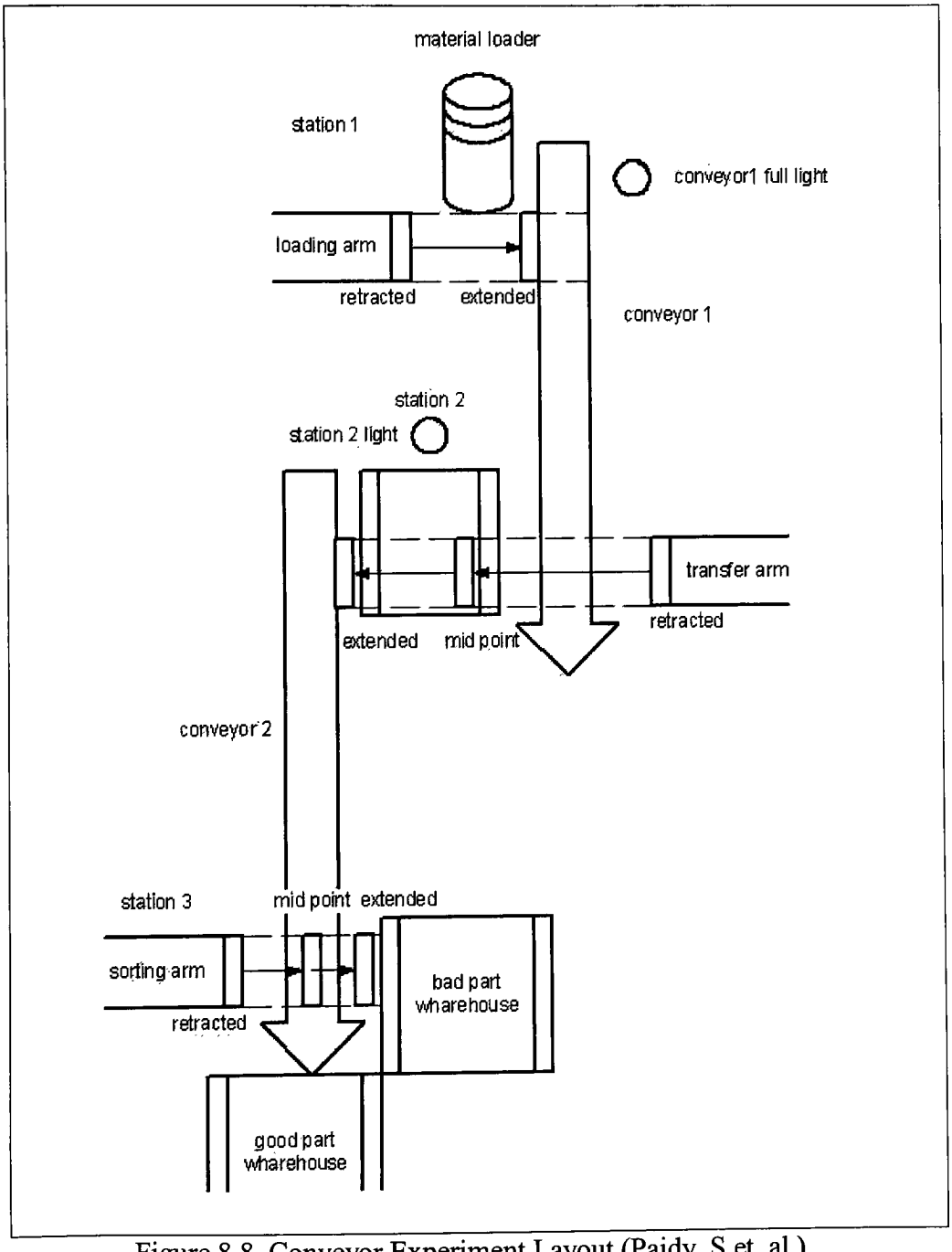


Figure 8.8, Conveyor Experiment Layout (Paidy, S et. al.)

The conveyor experiment as seen from Figure 8.8 has three stations namely the loading station (Station 1), transfer and service station (Station 2), and the sorting station (Station 3) respectively.

Station 1 consists of a loading arm that enters parts onto conveyor 1; it consists of a programming block that computes the arrival of parts and the type of parts (Part X, Y, and Z) that enters the system through probability distribution. The arrival distribution is set in a way that parts arrive into the system with a minimum of 5 seconds apart. It consists of two positions – the advanced and retracted position. The advanced position is used to place the part on conveyor 1, after which it is retracted. The station 1 task sends a message to station 2 indicating the arrival of the part with the information of the part included in the message.

Station 2 consists of the transfer arm and a service area. The transfer arm transfers the part present on conveyor 2 on to the service area. The presence of a part in the service area is sensed by a photo sensor, the time spend by the part in the service area depends on the part type (Part X, Y or Z), a message is sent station 3 indicating whether the part is a good or bad part. At this time the transfer arm is in its intermediary position. There is an additional photo sensor placed in such a way that it detects the presence of parts on conveyor 1 for congestion. The moment the sensor goes high, the task running station 2 sends a message to station 1 indicating it to stop the loading of parts on to conveyor 1 till the sensor goes low. After completing the amount of time spent in the service are the transfer arm pushes the part towards conveyor 2 by moving to its extended position. After the part is placed on to conveyor 2 the transfer arm comes back to its retracted position.

Station 3 consists of a sorting arm, which has 3 positions (retracted, mid point, and extended). Depending on the message received by the task running station 2, the sorting arm pushes the part to the bad part warehouse by moving to its extended position or moves back to its mid point position in order to let the part to be placed into the good part warehouse. Counters are used here to keep track of or keep a count of each type of parts (X, Y, and Z) as well as to keep a count of the number of good and bad parts in each part

type. Timers are also incorporated in the program to keep track of arrival times and service times.

8.4.2 Inputs/Outputs:

Shown in figure 8.9 are the inputs/outputs available for the conveyor experiment.

	Symbol	Address	Data type	Comment
1	Conveyor 1 On/Off	Q 0.0	BOOL	Controls Conveyor 1
2	Conveyor 1 Photo Sensor	I 1.5	BOOL	Sensor sensing allowed number of parts on conveyor 1
3	Conveyor 2 On/Off	Q 1.1	BOOL	Controls Conveyor 2
4	Conveyor1 full indicator	Q 2.0	BOOL	Indicates the full status of parts on Conveyor 1
5	Loading Arm Backward	Q 1.5	BOOL	Output to movethe loading arm in the reverse direction
6	Loading Arm Forward	Q 1.4	BOOL	Output to move the loading arm forward
7	Service Area Indicator	Q 1.3	BOOL	Controls the service area light
8	Service Area Sensor	I 1.6	BOOL	Sensor sensing the presence of part in the service area
9	Sort Arm backward	Q 0.2	BOOL	Output to move the soting arm backward
10	Sort Arm forward	Q 0.1	BOOL	Output to move the sorting arm forward
11	Transfer Arm backward	Q 1.7	BOOL	Output to move the transfer arm backward
12	Transfer Arm forward	Q 1.6	BOOL	Output to move the transfer arm forward

Figure 8.9, Inputs and Outputs for the Conveyor Experiment

There are two output lines for switching On/Off the two conveyors. Two output lines to turn On/Off the bulbs stationed at the service area (indicating presence of part in the service area) and on conveyor 1 (indicating the full status of parts on conveyor 1). There are 3 output lines to turn the 3 motors running the 3 arms in one direction (forward) and three output lines to turn the motors running the 3 arms in another direction (reverse).

There are two input lines that come from the experiment to the WinAC S7 program. One input line (photo sensor) indicates the presence of a part in the service area. The second input line indicates the S7 program the full status of the conveyor 1.

Other symbols are used that gives the program the information of parts flowing through the system, such as the total number of parts that would be served, number of good and bad parts for each type, the service times of each part and also the quality information. This information is stored in integer, real, and character data type format into memory word, memory byte, and memory double word that are basically the outputs of various counters and timers of the S7 Ladder logic program into a location of the PLC memory. The symbolic table containing this information is shown in Figure 8.10.

13	Transfer Arm Midpoint	MB	10	CHAR	Indicates the midpoint position of the Transfer Arm
14	Transfer Arm Retracted	MB	12	CHAR	Indicates the retraction of the Transfer Arm
15	Arrival Time	MD	10	REAL	Arrival time indicator
16	Begin Service	MD	14	REAL	Indicates time taken to begin service
17	Loading Arm Retracted	MB	18	CHAR	Indicates the retraction of the Loading Arm
18	Loading Arm Extended	MB	16	CHAR	Indicates the extension of the Loading Arm
19	No of parts completed	MW	30	INT	Number of parts out of the system
20	No of parts on Coneyor 1	MW	34	INT	Indicates the number of parts on the conveyor waiting to be serviced
21	Number of bad X parts	MW	14	INT	Counts the number of bad X parts
22	Number of bad Y parts	MW	20	INT	Counts the number of bad Y parts
23	Number of bad Z parts	MW	26	INT	Counts the number of bad Z parts
24	Number of good X parts	MW	12	INT	Counts the number of good X parts
25	Number of good Y parts	MW	18	INT	Counts the number of good Y parts
26	Number of good Z parts	MW	24	INT	Counts the number of good Z parts
27	Number of X Parts	MW	10	INT	Counts the number of X parts
28	Number of Y Parts	MW	16	INT	Counts the number of Y parts
29	Number of Z Parts	MW	22	INT	Counts the number of Z parts
30	Part No being serviced	MW	32	INT	Part being serviced in the service area
31	Quality	MB	26	CHAR	Indicates the quality of a part after passing the service area
32	Service time	MD	12	REAL	Logs service time for a part
33	Sorting Arm Extended	MB	20	CHAR	Indicates the extension of the Sorting Arm
34	Sorting Arm Midpoint	MB	22	CHAR	Indicates the midpoint position of the Sorting Arm
35	Sorting Arm Retracted	MB	24	CHAR	Indicates the retraction of the Sorting Arm
36	Total Number of Parts	MW	28	INT	Total Number of parts to be processed
37	Transfer Arm Extended	MB	14	CHAR	Indicates the extension of the Transfer Arm

Figure 8.10, Symbols of memory locations

8.4.3 Siemens and LabVIEW Integration through OPC for Conveyor Experiment:

The conveyor experiment's process monitoring and control is performed by the Step7 Ladder Logic programming. In order to make the inputs and outputs available for other applications, a symbolic table as shown in Figures 8.9 and 8.10 is created.

These symbols are then converted to tags by using the tag file configurator. A connection to the saved tag file is made, by taking the services provided by the computing configuration software.

The LabVIEW data acquisition software is employed to create the human machine interface that is shown in Figure 8.11.

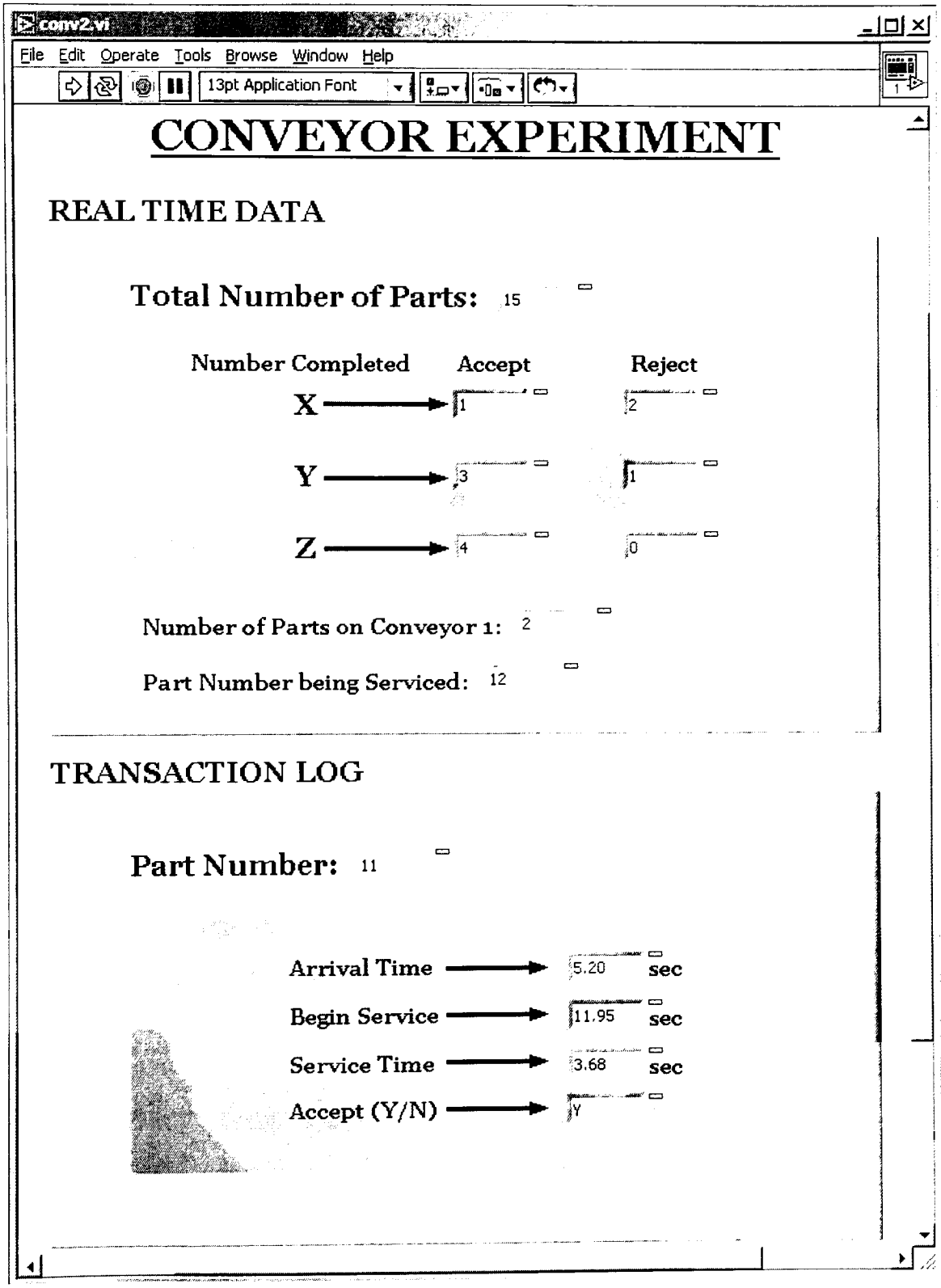


Figure 8.11, LabVIEW interface for the Conveyor Experiment

Figure 8.11 shows the human machine interface for the Conveyor Experiment. It consists of real time display of data for each part that enters the system. The interface also shows the statistics obtained from various counters to convey the number of each type of parts and also the good and bad parts for each of the part type. The transaction log or the detail for each part is also depicted in the interface.

In this application the tags are referenced or subscribed by the LabVIEW application from a remote computer on the same network as the computer in which the control program for the conveyor experiment resides. The tags are created in 'IMERT78', a computer in the laboratory, while the LabVIEW application runs in a remote computer.

DataSocket Transfer Protocol is used by LabVIEW to access the tags that are placed in the remote computer, an example of one of these tags is shown in Figures 8.12a and 8.2b.

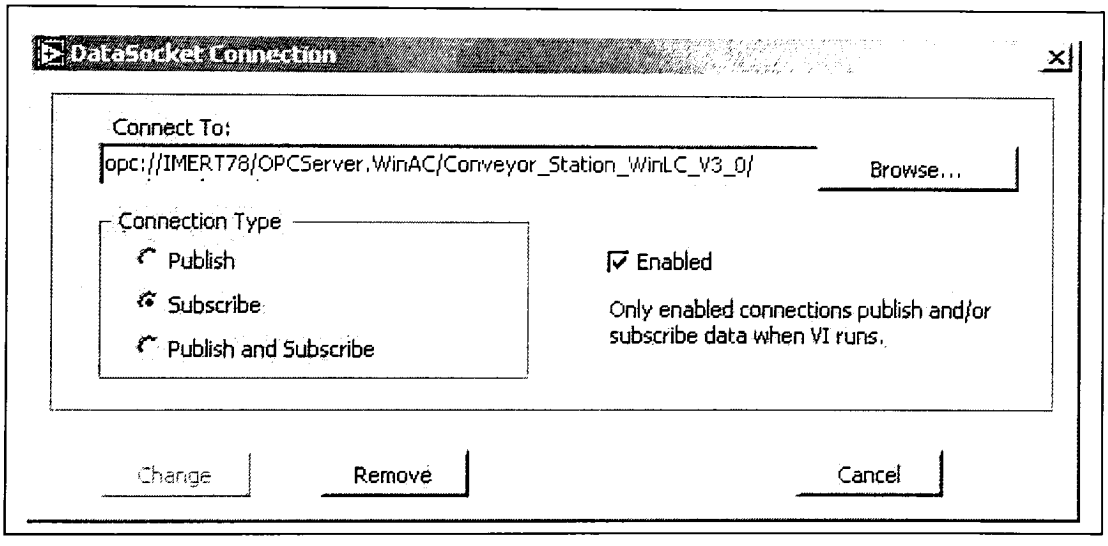


Figure 8.12a, Connection for "Number of good X parts" Tag (LabVIEW6i Development System)

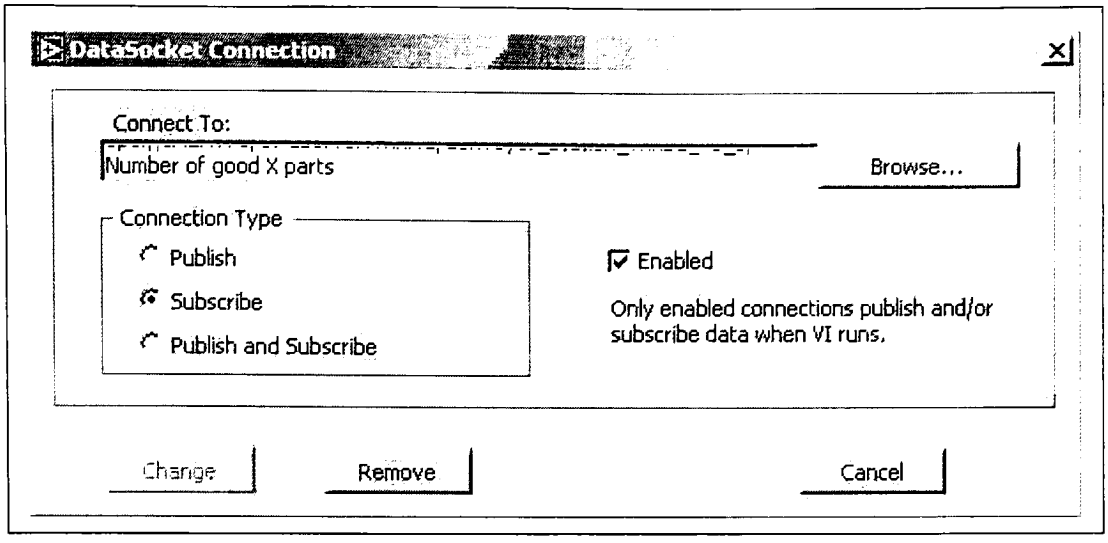


Figure 8.12b, Connection for "Number of good X parts" Tag (LabVIEW6i Development System)

Figures 8.12a and 8.12b shows the DataSocket connection being made for the "Number of good X parts" tag present in computer "IMERT78" using OPC technology. Thus we see that if the random distribution in the service area task computes the X part to be bad, the count of bad X parts in the memory location is incremented by the counter, the tags associated with the memory location would give a new value to the indicator (residing in the LabVIEW interface) that subscribes to it from over the network, so that the tag value (thus the memory value) could be depicted in the human machine interface application created in LabVIEW.

Reference:

1. Paidy, S., Reeve, R. – "The Use and Development of Physical Simulators", Annual Simulation Symposium, May 1980.

9. Conclusion

In this thesis study we reviewed various data acquisition networks available in the market and the inconsistencies in their architecture to provide integration to a number devices available for process control and automation. We also looked at various drawbacks introduced by different data acquisition or fieldbus networks, when it came to integrating "islands of automation" existent in today's manufacturing scenario. This was mainly due to the various protocols followed by the various networks in their physical layer (Layer 1 of the OSI Model) that prevented devices from being interoperable between different networks.

In order to prevent the amount of time spent by device manufacturers in developing device drivers for each network that it needs to be connected with, and the time needed to develop device-specific code for every application that needs to be connected to the number of devices on the factory floor, we looked at two technologies that could help alleviate the "driver" problem that is prevailing in today's world.

The first technology discussed was the DataSocket Transfer Protocol developed by National Instruments in their LabVIEW software that facilitated data sharing between two applications that run in a single machine or two machines in a network. We introduced the DataSocket technology to demonstrate the client-server model followed in ensuring smooth integration of data between publishers (source of data) and subscribers (clients for the published data) in a network. Focus has been given to integrating data between applications running on different computers in a network in order to demonstrate the capabilities of the technology in an industrial setting, since data is shared between applications that reside in different levels of the Process Control Information Architecture. These levels are connected in the same network in order to allow smooth data sharing and company wide integration; the physical media used in each of these levels would vary, which brought the need for the second technology.

OLE for Process Control (OPC) is a technology developed by Microsoft that is based on Component Object Model (COM)/ Distributed Component Object Model (DCOM)

software components. The change in the physical layer protocols at various levels of the Process Control Information Architecture forces us to develop drivers to obtain data from data sources. OPC does away with this need by only needing OPC interfaces between the server (data source) and the client (application needing the data from the server). There are a number of manufacturers that identify the need of their devices being more and more interoperable with the available networks, which has forced them to accept this technology and thereby reduce the time needed for customization. This technology allows manufacturers to focus their attention towards developing newer and better automation devices, instead of wasting their efforts towards developing drivers for available networks and protocols.

The only drawback of this technology is that by accepting or embracing OPC technology (platform), manufacturers tend to lose the platform independency as one is forced to implement Microsoft based software and computing methods. With the PC getting more and more robust by qualifying the set industry standards and also by finding its way to the factory floor introducing the concept of PC-based control and automation, the need of manufacturers to move to different platforms is greatly reduced.

On the other hand with OPC hardware manufacturers have to make only one set of server software components, to which customer's client software can connect and exchange data on a very consistent way. Thus in future OPC could be a dominant standard interface between different automation system vendors. This was demonstrated by the application that integrated two different systems (Siemens Simatic PLC and LabVIEW) by using the OPC interfaces that exist between them.

Appendix I

Integration of NI LabVIEW with Siemens Simatic PLC

Integration of NI LabVIEW with Siemens Simatic PLC

The following steps are to be followed to enable data access between Siemens PLC and LabVIEW client application interface using OPC technology:

1. Start WinLC Controller by clicking on its icon on the desktop.
2. Choose the 'Run-P' button on the WinLC interface in order to run in the PLC in the Run-P mode.
3. Now double click on the Simatic Manager icon in order to start Step 7
4. Create a new project, for example "OPCAPP" and click 'OK' to open this project.
5. Insert a new Simatic PC station (Insert → Station → Simatic PC Station)
6. Double click on the Simatic PC Station to open a window showing the icon for Configuration.
7. Double click on the Configuration icon to open the window for Hardware Configuration called the 'HWConfig'.
8. In the Station menu of the HWConfig window select 'Import' in order to import a '.cfg' or a configuration file that consists of the hardware configuration information for the PLC.
9. Importing the '.cfg' file would also import the symbols (symbolic names given to the I/Os), if any, contained in it.
10. Download the configuration on to the ET200M module in order to complete the hardware configuration.
11. Now switch over to the Simatic Manager window and drop the 'S7Program(1)' menu within the project to reveal S7 source files (Sources) and organizational blocks (Blocks) for the project.
12. Click on 'Blocks' to reveal the available organizational blocks (OB) on the right window.
13. Double click on the 'OB1' icon to open 'LAD/STL/FBD' window that would allow you to create a PLC program in any of the 3 formats namely, Ladder Diagram (LAD), Statement List (STL), and Functional Block Diagram (FBD).
14. Create a program using any of the 3 formats, using the symbolic names (names corresponding to the I/Os in the module) for the inputs and outputs.
15. Save and compile the program and download it to the ET200M module.

16. Run the program either 'online' or locally (without the monitoring capability from Simatic Manager), to check for the functionality of the program to realize the required control objectives.
17. Now select the 'Computing Tagfile Configurator' option by following the 'Start → Simatic → PC based Control' path. This step is performed in order to create OPC tags from the existing symbols contained in a program.
18. In the 'Tag File Configurator' window, click on the 'Program...' option from the 'Insert' drop down menu.
19. On performing the above step, a child window titled 'SIMATIC Program(s)' is called. This window shows the programs or projects resident on the local computer.
20. Select the sample program "OPCAPP" and drop down right to the level where 'S7 Program(1)' can be found for the selected project.
21. Click on the arrow button with the'S7 Program(1)' highlighted, in order to move the highlighted 'S7 Program(1)' to the right, within the child window and select 'OK'.
22. This would transfer the symbols resident within the program to OPC tags or create OPC tags for the corresponding symbols resident within the program.
23. Now click on the 'Save' button on the resulting window, in order to save the Tag Source Data File into '.tsd' format.
24. Now select the 'Computing Configuration' option by following the 'Start → Simatic → PC based Control' path. This step is performed in order to allow client applications (like LabVIEW) to access the OPC tags contained in the '.tsd' file created in the previous step.
25. The connection to the tags is established by browsing the just created '.tsd' file in the 'Connection via Tag Source' option.
26. Clicking 'OK' would now create an OPC interface that would allow client applications to access data generated by the program running on the Siemens PLC.
27. Now start LabVIEW program and create a new VI.

28. Insert the required controls and indicators on the Front Panel that would symbolize a Human Machine Interface.
29. Right click on the control or indicator and select the 'Data Socket Connection.....' option from the Data Operations sub menu.
30. In the Data Socket Connection dialog box, click browse and select 'Browse Measurement Data....'. This selection would browse the local computer as well as the network for available publishable and subscribable tags.
31. Since the tags for the application were created on the local computer, click on 'OPCServer.WinAC' under the 'My Computer' selection. Click on 'Simatic_PC_Station(1)_WinLC)V3_0' to reveal the tags contained within it. If the tags are available from a remote computer connected within the network, then click on 'My Network Places' and locate the computer on which the tags would reside.
32. Select the required tag that you would want to associate the control or indicator with and click 'OK'.
33. Now in the 'DataSocket Connection' dialog box, one could see the URL to the specified tag resident on the local or the networked computer (e.g. opc://localhost/OPCServer.WinAC/SIMATIC_PC_Station(1)_WinLC_V3_0/Sim Input 7).
34. Select either 'Publish' (if you want to write data values to the tag), 'Subscribe' (if you want to read data values from the tag), or 'Publish and Subscribe' (if you want to both write as well as read values to or from the tags).
35. Click on 'Connect' to establish a connection with the selected tag.
36. Perform steps 29 to 35 to repeat the process in order to establish a connection between the OPC tag and a control or an indicator on the Front Panel.
37. Perform the required logic on the Block Diagram and then run the LabVIEW program after saving it to a known location.
38. If the connection was set correctly, one could see that the LabVIEW program would be able to acquire data from the process running under Siemens and also control the process if desired.

Thus, OPC technology is used here to enable smooth integration between fundamentally different systems and thus aid in transferring process related data from the Field Management (Siemens) Layer to the Process Management (LabVIEW) Layer in the Process Control Information Architecture.

Appendix II

Using CP 5613 with National Instruments OPC Clients

Using CP 5613 with National Instruments OPC Clients.

By Carlos Barberino

Requirements:
Driver: DP-5613/Windows NT 4.0
DP-OPC Server/ Windows NT 4.0
Com PROFIBUS V5.0

Configuration:
The first step is configuring the DP Master and Slave network. To do this *SIEMENS* has an easy to use interface, called COM PROFIBUS. This interface looks like Windows Explorer with a drag and drop feature to configure your DP networks. Find your DP master device, click and hold with the left mouse button, drag that and drop into the white area at the right hand side. Find your DP slave as well and do the same thing.

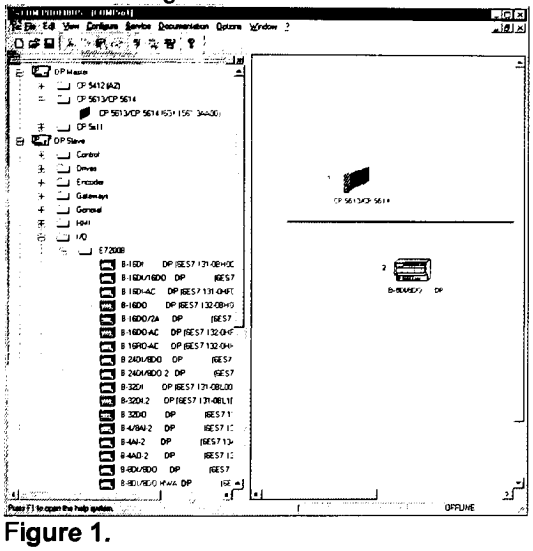


Figure 1.

If you move your cursor over the icon you can see the parameters for each device. If you right click on your device you can choose Properties and change the slave address as well as the parameters for that device.
When you are done with the configuration you can save it in a file with extension **pb5** as you can see in Figure 2.

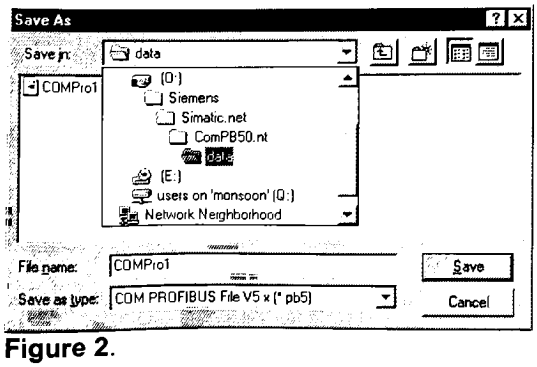


Figure 2.

You can create a database, which is going to be used later on, by exporting the NCM file. Neither the getting started manual nor the online help makes clear what NCM stands for and its definition, which is included in the Com PROFIBUS glossary, says that it is "a
File (with the extension .LDB) that contains all the parameters created with
COM PROFIBUS for a master system. This file is required as a binary database for the SIMATIC NET PC modules".
Then go to ***File >> Export >> NCM File*** and you will see the following window.

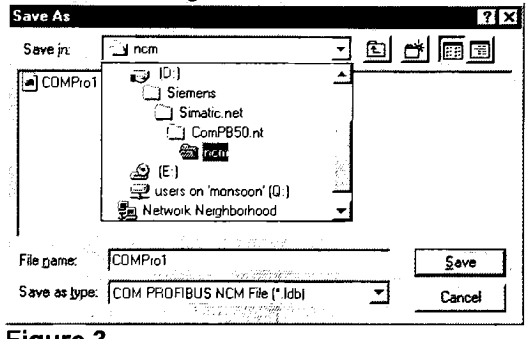


Figure 3.

This database is going to be used by the PG configurator and the OPC server as well.
The master and slaves have to be configured in the ***Simatic > Simatic Net > Set PG/PC Interface***. Pick the ***CP5613_5614 (PROFIBUS)*** and hit the ***Diagnostics*** pushbutton

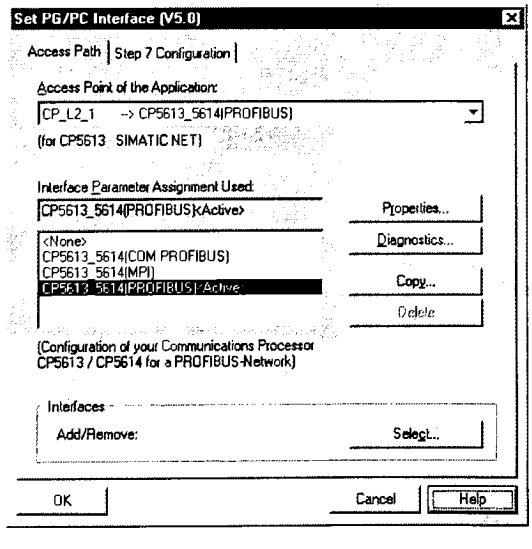


Figure 4.

Double click on ***CP5613_5614 PROFIBUS***.

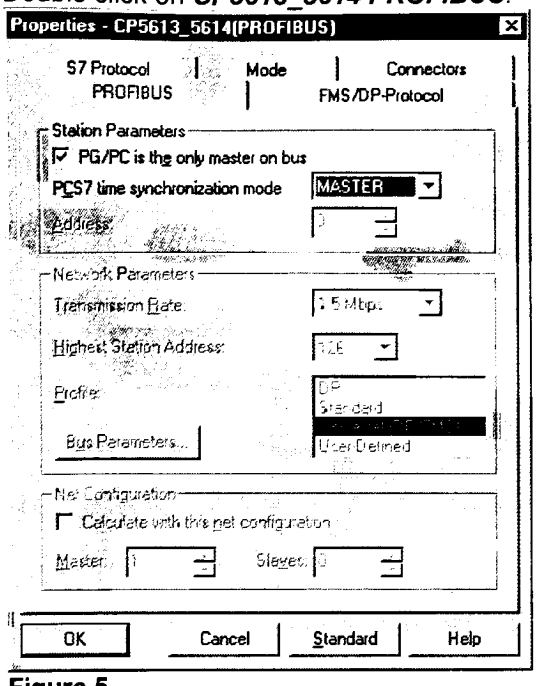


Figure 5.

Click on the ***PROFIBUS*** tab and check ***"PG/PC is the only master on bus"***. Choose master for the PCS7 time synchronization. Click on the FMS/DP-Protocol tab.

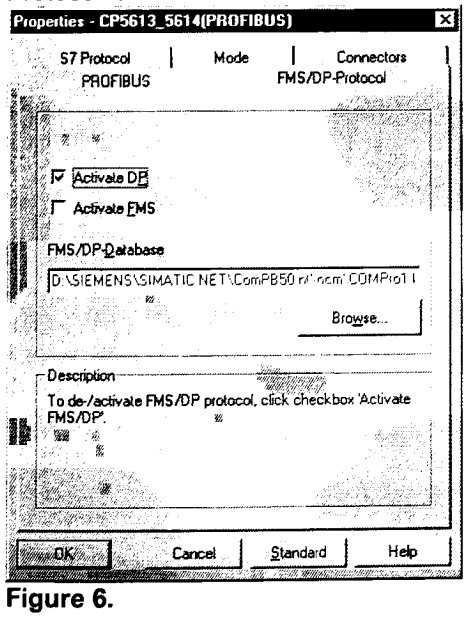


Figure 6.

Check the ***Activate DP*** check box and browse the ***LDB*** file you exported from Com PROFIBUS. Hit Ok. Click on the ***Mode*** tab and you should be able to reset, restart and test the PG interface.

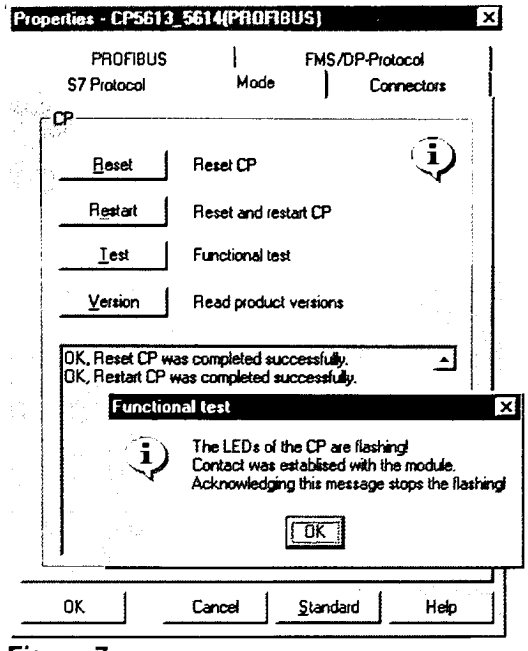


Figure 7.

Hit ***Ok*** to close the utility.
Once you are done with that you should be able to see the PROFIBUS items in any OPC Client such as **OPC Scout** or **National Instruments OPC Clients**.

Using Server Explorer

In Server Explorer for instance you could pick ***OPC SimaticNET***, which should show up in your server list if it was installed and registered properly in your computer.

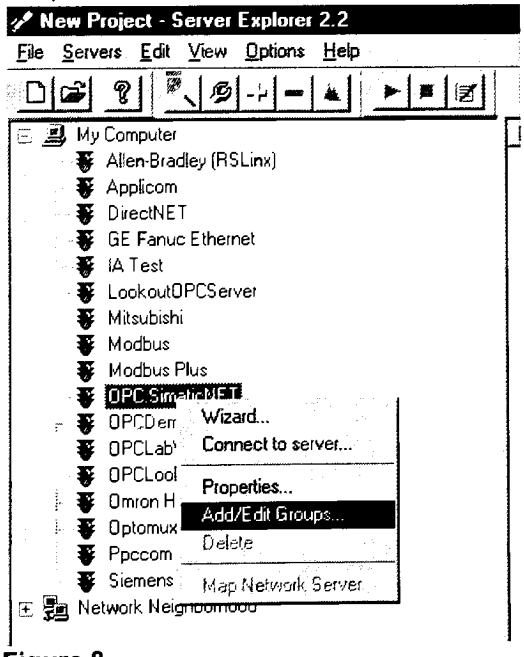


Figure 8.

Create a new group, connect to the server and create items. You should see the items corresponding to your I/Os

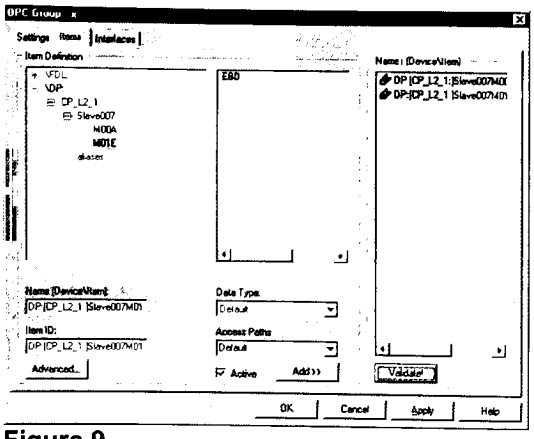


Figure 9.

Add the channels you are interested in, click **Add**, select all the elements in your list and validate it. Click ***Ok***. You should see the channels in the right hand side of your screen.
At this point you should be able to read from and write to your devices.

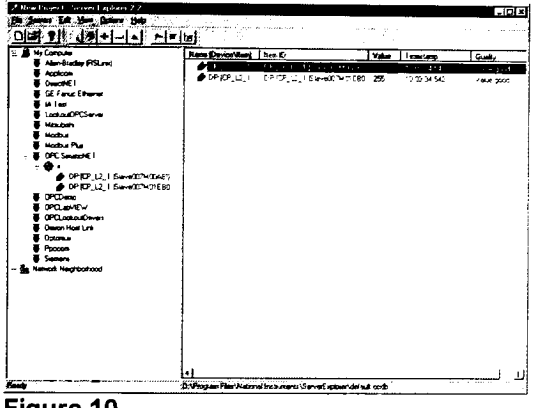


Figure 10.

Using Lookout

In Lookout you could use the ***OPC Client*** object to communicate with you DP device. Open Lookout, create a new panel and go to ***Object >> Create***. You should see a list box with a folder called drivers. Click on that and choose ***OPC Client***.
Select the process where you want to create this object and fill the configuration window as follows.

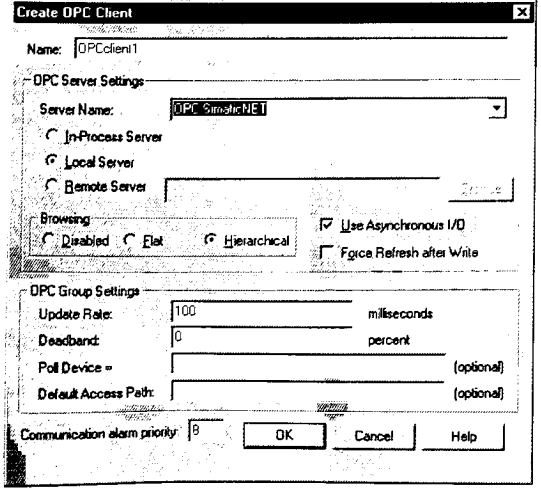


Figure 11.

If you open the ***Object Explorer*** you should see all the items defined for this particular server.

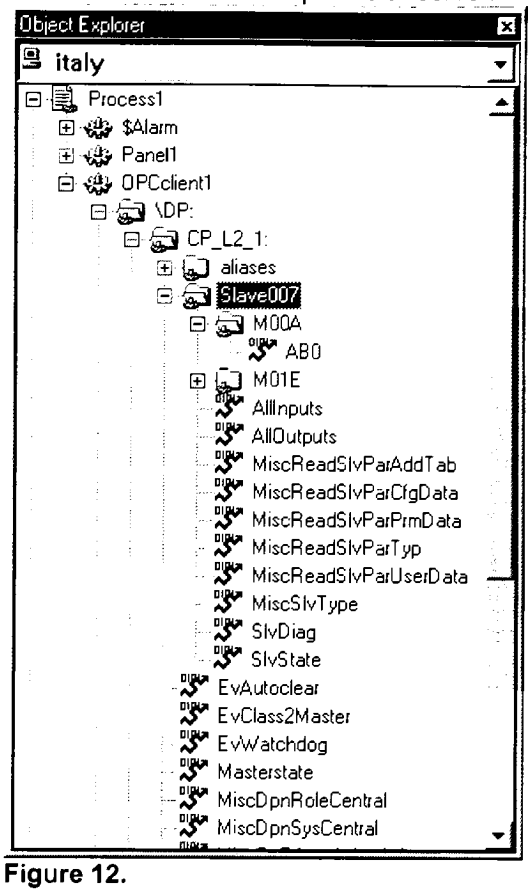


Figure 12.

Drag and drop the item you want into your panel. At this point you should be able to read data from your server.

Using LabVIEW Dataloging and Supervisory Control Toolkit

Create a new VI. In this example I can write an analog value to the DP device. You might need to use a digital though, so in this case your program could be slightly different than this one. Once we are using DataSocket to communicate with the OPC Server, I included a 20ms delay, otherwise the client would try to read data as fast as possible from the server using a lot of CPU time. So you should include at least 2ms delay in your while loop.

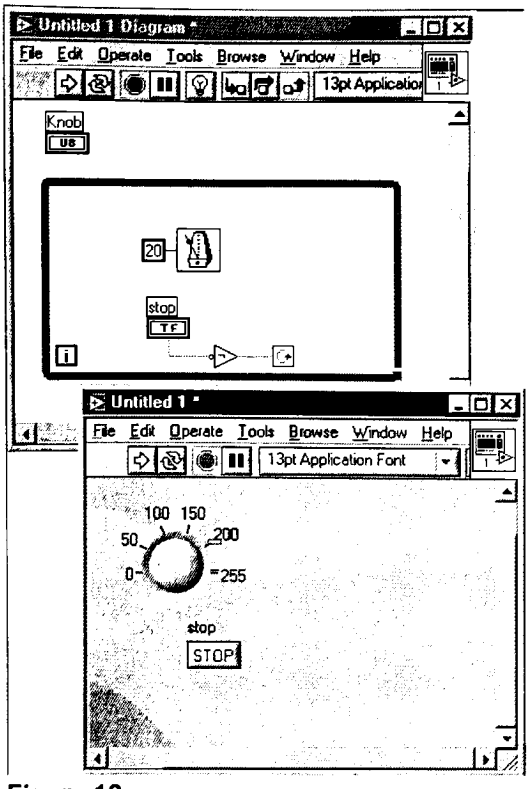


Figure 13.

Right click on your control and choose ***Data Operations > DataSocket Connection***.

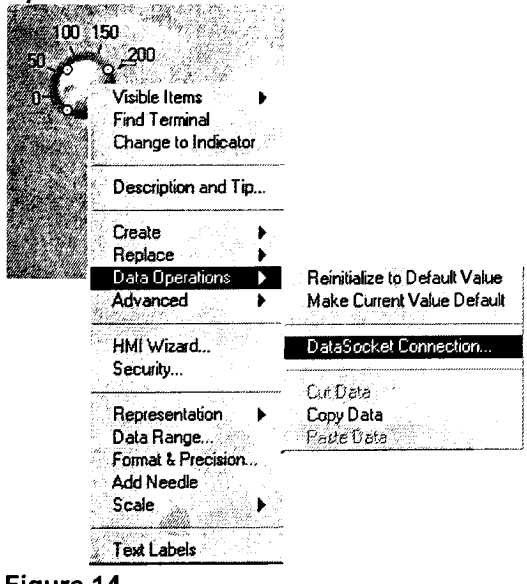


Figure 14.

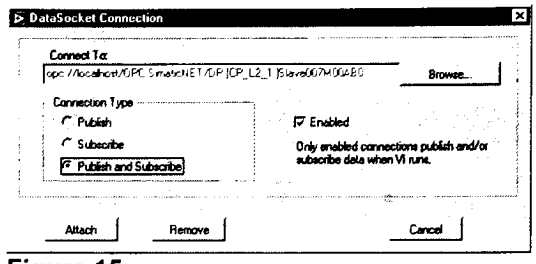


Figure 15.

Click ***Browse*** and in DataSocket Browser choose ***OPC SimaticNET*** and the item you want to read from or write to. If you are controlling an output you have to check the ***Publish and Subscribe*** check box. Click the **Attach** button and run your VI.