MIXED-MODE SCADA NETWORK FOR HYDRO ELECTRIC POWER PLANT

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

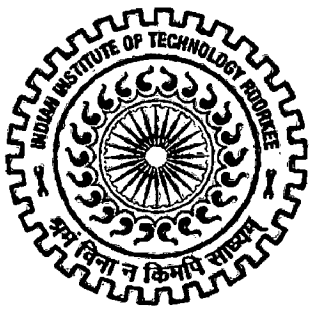
in

ELECTRICAL ENGINEERING

(With Specialization in Measurements and Instrumentation)

By

N. SAI MANOHAR



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)
JUNE, 2008

I.D. No.-M.T. 446 /2008-30 HKV-RPM

CANDIDATE'S DECLARATION

I hereby declare that the work which is being presented in the dissertation entitled **"MIXED-MODE SCADA NETWORK FOR HYDRO ELECTRIC POWER PLANT"** in partial fulfillment of the requirements for the award of the degree of Master of technology in Electrical engineering with specialization in Measurements and Instrumentation, submitted to the Department of Electrical engineering, Indian Institute of Technology Roorkee, India is an authentic record of my own work carried out during a period from July 2007 to June 2008 under the guidance and supervision of **Dr. H.K. VERMA**, Professor and **Dr.R.P. MAHESHWARI**, Professor, Electrical Engineering department, Indian Institute of Technology, Roorkee.

The matter presented in this project has not been submitted by me for the award of any other degree of this Institute or any other Institute.

Dated: JUNE-30, 2008

Place: Roorkee

N. Sai Manohar

(N. SAI MANOHAR)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

30/6/08

Dr.H.K.VERMA

Professor,

Department of Electrical Engineering,

Indian Institute of Technology Roorkee,

Roorkee-247667

30/06/08

Dr.R.P. MAHESHWARI

Professor,

Department of Electrical Engineering,

Indian Institute of Technology Roorkee,

Roorkee-247667

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to **Dr. H.K.Verma, Professor** and **Dr.R.P.Maheshwari, Proffessor,** Electrical Engineering Department, Indian Institute of Technology Roorkee, Roorkee, for providing me all the necessary guidance and inspirational support throughout this dissertation work. I am grateful to **Dr.H.K.Verma** for giving me the opportunity to work on the Foundation Fieldbus and MODBUS equipment.

I would also thank Dr.R.S.Anand, Associate Professor, Electrical Engineering Department for his immense support.

My heartfelt gratitude and indebtness to all the teachers of Measurements and Instrumentation group who, with their encouraging and caring words, constructive criticism and suggestions have contributed directly or indirectly in a significant way towards completion of this project.

I am indebted to the staff of M&I lab, all my friends and family for their encouragement and support which made the completion of this dissertation possible.

ABSTRACT

The hydro power generation is one of the efficient and economic forms of all the renewable energy sources. Hydro power stations are located at remote places and require an automation system which would continuously monitor various plant parameters and remotely control operations of the plant.

SCADA (Supervisory Control and Data Acquisition system) performs the function of data acquisition, for providing an unattended, efficient and reliable supervision and control. SCADA network includes interconnection with the station computers and device level networks. At the device level, SCADA employs standard protocols for communication with the field devices and several clients connect to the work station server to access the parameters for display and processing.

In the past few years, industry has developed at the level of the field-devices at its own range of digital communication protocols with the increasing involvement of intelligent instrumentation. When these networks are used in a plant it requires access to the plant parameters for the remote control and centralized supervision. The applications had to develop drivers for their own packages to access the data. Hence there was an urgent need to develop standard schemes for communicating with multiple protocol networks.

A mixed-mode SCADA network is one which can connect the work station to a number of device level networks and provide data to multiple client applications simultaneously. In the present work a mixed-mode SCADA network, integrating a FOUNDATION FIELDBUS (FF) and two RS485/MODBUS networks, is implemented for a model hydro power station of 2* 1.5 MW capacity. The network has a central server node connected to the Fieldbus network through a bridge (Ethernet to FF) and to two Modbus networks through RS232 to 485 convertor. Each device level network has its individual OPC server (Object Linking and Embedding for Process Control) and acts as an interface between hardware providers and software developers. It provides a mechanism to provide data from a data source and to communicate the data to any client application in a standard way. The OPC servers installed on the server node are Dfi OLE server, NAPOPC server and NDS OPC server. Dfi Ole Server acquires the real-time data

from the Fieldbus devices where as NAPOPC and NDSOPC servers have been used to configure and acquire the data from the Modbus networks. Fieldbus devices have been used to measure the pressure heads at the forebay and tailrace, water level, inlet nozzle position, various generator temperature and current. The Modbus devices monitor generator and transformer voltages, currents, temperatures, frequency and the breaker and isolator positions. GraphworX, AlarmworX, TrendworX, LabVIEW and Visual Basic have been used as the OPC clients. GraphWorX is an OPC compliant human-machine interface (HMI) software package for process control. AlarmWorX is OPC-compliant alarming software, performs alarm detection and reporting based on the OPC Alarm and Events Standard. TrendWorX is similar to GraphWorx but is equipped with more advanced options such as different kinds of plots etc. Smar WebHMI has been used as a thin client Web solution that enables standard Web browsers, such as Microsoft Internet Explorer, for use as real-time operator interfaces.

In the present work two remote clients are connected to the plant server on local area network (LAN). The clients have been configured with DCOM (Distributed Component Object Model) and windows registry settings to communicate the installed OPC clients with the servers on the central server node so that any client can monitor and control.

Thus a mixed-mode SCADA network involving Foundation Fieldbus, Modbus and Ethernet protocols has been successfully implemented in the laboratory with the above functionalities.

INDEX

LIST OF FIGURES

CHAPTER-1 INTRODUCTION

1.1 SCADA Networks

SCADA stands for Supervisory Control And Data Acquisition system. A SCADA system comprises of several field devices like PLCs, transmitters, sensors and actuators for measuring and controlling the field parameters. PLCs and Distributed Control Systems (DCS) are now-a-days offering Ethernet/TCP/IP connectivity so that the real-time information on the plant can be readily accessible by any work station on the plant network (LAN/WAN), in a client-server relationship. A plant manager can visualize the graphical display of the real-time values of the parameters in the plant floor and all the plant operations in a single window. Industrial plants are using information technology for the years, but with open IT standards, faster computers, emerging software and demand for integrated information by all segments of an industrial enterprise (operations, maintenance and management), industrial networks are merging right into enterprise-wide IT solutions.

Meanwhile at the device level, industry has developed in its own range of field bus or device-level networks for linking control devices with increasingly intelligent instrumentation. The field devices are also becoming more complex, smarter and hold more information about themselves and the processes to which they are connected. A higher communication data rates are required to handle this additional information without undue delays. Hence several digital communication protocols have emerged out for connecting plant network with the field device networks.

1.2 Mixed-mode SCADA Network

There are many sorts of device level buses coexisting in the industrial control systems but they cannot communicate with each other because of different protocols used in these buses. In a plant utility normally there are different buses utilized in systems, in which information communication with each other is often needed. Therefore, scheme of different

kind of bus network system integration that can realize data communication between these networks of different protocols must be investigated. Such a system for SCADA applications can be called a "Mixed-mode SCADA network". A standard scheme must be used to the SCADA network an open system and interoperable for any network protocol. The ideal opposite to proprietary system is an open system. Open systems are based on off-the-shelf standards enabling multiple vendors to provide interoperable hardware and software. This scheme enables the user to choose manufacturer of the instruments freely without any sticking-up with a single manufacturer, thereby selecting the devices with latest technology and features, irrespective of type of the existing devices and instrumentation.

1.3 Statement of Problem

The statement of the problem is described as below. This can be split into smaller tasks that can be well understood.

- Investigate the various available schemes for the integration of different protocol buses in a SCADA network and select a best and standard scheme for implementation.
- First to configure Foundation fieldbus network and Modbus network individually, and then integrate these two networks into a single network connected to a single server station. Thereby to develop a network such that multiple client applications installed on remote nodes can access real-time data from the server station on LAN.
- To develop a remote client application to access real-time data from the server of both protocol networks viz. Foundation fieldbus and Modbus.
- Develop the web service by installing a web server on the server station and to publish web pages so that the HMI windows can viewed as web pages through web browsers on LAN

1.4 Organization of Dissertation

Chapter 1 includes a brief introduction about the SCADA networking concepts. It introduces the definition of the Mixed-mode network, its need for the process control applications and its benefits. The statement of problem of the dissertation is explained

In Chapter 2 three possible schemes for implementing a mixed-mode network Viz. Integration based on Gateway, Based on DDE (Dynamic Data Exchange) and Net DDE and that Based on OPC (OLE for Process Control) are discussed. The three schemes are compared for their performance and one of the schemes is selected for implementation.

The chapter 3 discusses in detail about the features of an OPC based SCADA network, it further explains in detail about the technologies used by an OPC network. Several OPC standards given by OPC foundation group are discussed in detail in this chapter. It also discusses the client-server architecture of an OPC based network, client-server communications and structure of an OPC Server and client as well.

Chapter 4 discusses in detail of the field instruments of Modbus and Foundation Fieldbus used for SCADA system. It then discusses of the software used for the configuration of the field devices, client software used to access the real-time data from the servers. It also discusses in detail the procedure for developing the OPC client applications in Lab view and Visual basic as well.

Chapter 5 discusses the implementation of the above developed mixed-mode SCADA network as an application for small Hydro-power plant. It discusses in detail of the model Hydro power station, network architecture, SCADA functions and the details of the implementation of web-based monitoring.

Chapter 6 summarizes the various results and discussions. The results included in this chapter are those which cannot be provided as a part of the other chapters but have overall significance. These results also provide comparison with the other existing implementations if available.

Chapter 7 gives the Conclusions and Scope for Future work. This includes the concluding remarks and the other developments expected in various fields of applications. Some of the results, necessary network settings and Configuration details are also included in the Annexure, where a more detailed explanation is possible.

The References which have been cited are included.

2.1 Integration Based on Gateway [1][2]

The gateway is a computer system or other device which is used to interconnect two systems or networks using different communication protocols, data format structure, languages and system structures. It can be used to interconnect controlled network and information network in a SCADA network when they are working with different protocols from the network layer. Gateway accepts the packets coming from one network reformat it to the form the network understands and transmits it. . A gateway must understand the protocols used by each network linked into the router.

2.1.1 Architecture of Gateway based Network

The architecture of interconnected control networks and information through a gateway will be as shown in figure 2.1 as below.

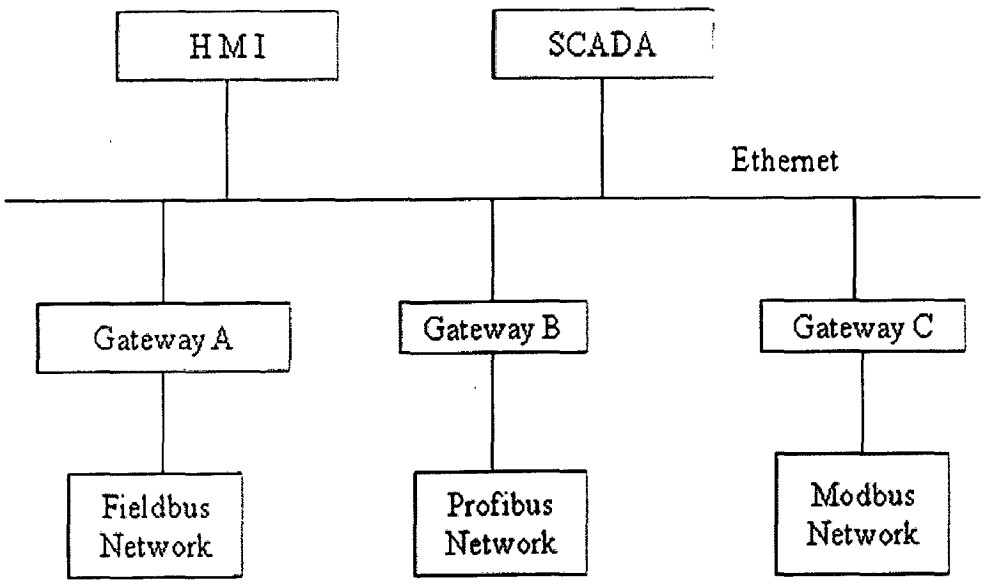


Fig. 2.1 Gateway network architecture

The Figure shows three device level networks of different protocols namely Foundation fieldbus, Modbus and Profibus, connected to the server and information network through their individual gateways. The gateways integrate the three networks through

protocol transits. There are two methods to transit bus protocols. One is to transit in two protocols that is bus protocol transit is one-one in any two protocols. The other is to choose a kind of bus protocol as a public protocol, then transit others to this object protocol and the bus protocol transit is many to one. A network gateway can be implemented completely in software, completely in hardware, or as a combination of both. It is an effective way to solve the problem of the system integration of the different bus protocols through the protocol transit.

At present, the Ethernet (TCP/IP) can be used as the best object protocol, because it is now widely used as intranet, it is becoming a most important network in the field of industrial automation. These days the Foundation Fieldbus has issued the High speed Ethernet as the new H2 norm and it has been voted for one of the 8 kinds of field buses in IEC61158, and the others are also developing the technology connecting to Ethernet. The devices in the factory can be tightly connected with the method of integration with the gateway and the enterprises will not depend on only certain products if we explore the transiting technology and products of multi-buses protocols.

However, there are so many kinds of fieldbus control networks, whose structures and features are of great differences. In the above figure 2.1, the Modbus uses RS 485 in its physical layer. So in order to make it integral part of the system it needs a different gateway which takes the data from RS485/Modbus network and reformats it to an Ethernet packet. If enterprise chooses many field buses, the system will become bigger as well as the price, which will make the gateway become the bottleneck of the whole system. At the same time, there are some problems to be solved to use the Ethernet as the industrial network and the most important one is the real time of the Ethernet. The Ethernet is a non-confirmed network based on collision detection and avoidance, so when the network flux is added, the responding time will be slow.

2.2 Integration based on DDE and NetDDE [1]

DDE stands for Dynamic Data Exchange. It is one of the Microsoft Windows methods for transferring data between applications. The DDE protocol is a set of messages and

guidelines. It sends messages between applications that share data and uses shared memory to exchange data between applications. Applications can use the DDE protocol for one-time data transfers and for continuous exchanges in which applications send updates to one another as new data becomes available.

2.2.1 Dynamic Data Exchange Protocol [33]

Windows has a message-based architecture, passing messages is the most appropriate method for automatically transferring information between applications. However, messages contain only two parameters (wParam and lParam) for passing data. As a result, these parameters must refer indirectly to other pieces of data when more than a few words of information pass between applications. The DDE protocol defines exactly how applications should use the wParam and lParam parameters to pass larger pieces of data by means of global atoms and shared memory handles. The DDE protocol has specific rules for allocating and deleting global atoms and shared memory objects. A global atom is a reference to a character string. In the DDE protocol, atoms identify the applications exchanging data, the nature of the data being exchanged, and the data items themselves

2.2.2 Dynamic Data Exchange Concepts [33]

Two applications participating in DDE are said to be engaged in a DDE conversation. The application that initiates the conversation is the DDE client application, the application that responds to the client is the DDE server application. An application can engage in several conversations at the same time, acting as the client in some and as the server in others. A DDE conversation takes place between two windows, one for each of the participating applications. A window may be the main window of the application or a hidden (invisible) window whose only purpose is to process DDE messages. Since a DDE conversation is identified by the pair of handles to the windows engaged in the conversation, no window should be engaged in more than one conversation with another window. Either the client application or the server application must provide a different window for each of its conversations with a particular server or client application. An application can ensure a pair of client and server windows is never involved in more than one conversation by creating a

hidden window for each conversation. The sole purpose of this window is to process DDE messages.

(a) Architecture of DDE Based Network [1][2]

The Fig 2.2 shows a schematic of the interconnected network of Foundation Fieldbus and Modbus networks with DDE Client-servers.

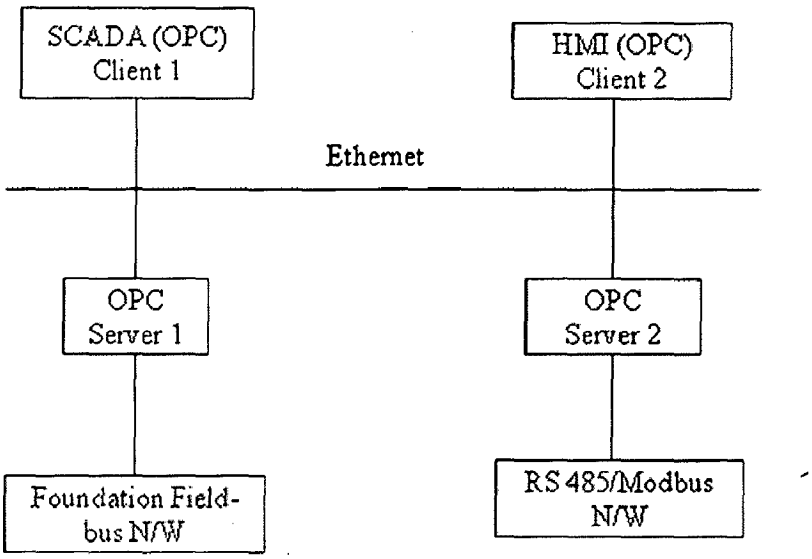


Fig 2.2 Architecture of DDE based Network

The server will control the communications on the device level network and polls the devices and holds the real-time values of the parameters. The applications of the information network such as HMI applications, Alarm & Event applications and data logging applications which access the real-time data form the clients.

(b) Application, Topic and Item Names

The DDE protocol identifies the units of data passed between the client and server with a three-level hierarchy of application, topic, and item names. Each DDE conversation is uniquely defined by the application name and topic. At the beginning of a DDE conversation, the client and server determine the application name and topic. The application name is usually the name of the server application. The DDE topic is a general classification of data within which multiple data items may be "discussed" (exchanged) during the conversation.

Because the client and server window handles together identify a DDE conversation, the application name and topic that define a conversation cannot be changed during the course of the conversation. A DDE data item is information related to the conversation topic exchanged between the applications. Values for the data item can be passed from the server to the client or from the client to the server. A special registered format named Link identifies an item in a DDE conversation.

(c) Permanent Data Links

Once a DDE conversation has begun, the client can establish one or more permanent data links with the server. A data link is a communications mechanism by which the server notifies the client whenever the value of a specified data item changes. The data link is permanent in the sense that this notification process continues until the data link or the DDE conversation itself is terminated. There are two kinds of permanent DDE data links warm and hot. In a warm data link, the server notifies the client that the value of the data item has changed, but the server does not send the data value to the client until the client requests it. In a hot data link, the server immediately sends the changed data value to the client.

(d) DDE Message Flow

A typical DDE conversation consists of the following events:

1. The client application initiates the conversation, and the server application responds.
2. The applications exchange data by any or all of the following methods:
    - The server application sends data to the client at the client's request.
    - The client application sends unsolicited data to the server application.
    - The client application requests the server application to notify the client whenever a data item changes (warm data link).
    - The client application requests the server application to send data whenever the data changes (hot data link).
    - The server application carries out a command at the client's request.
3. Either the client or server application terminates the conversation.

(e) Server Message Processing

Server applications respond according to the logic illustrated in the following fig.2.3

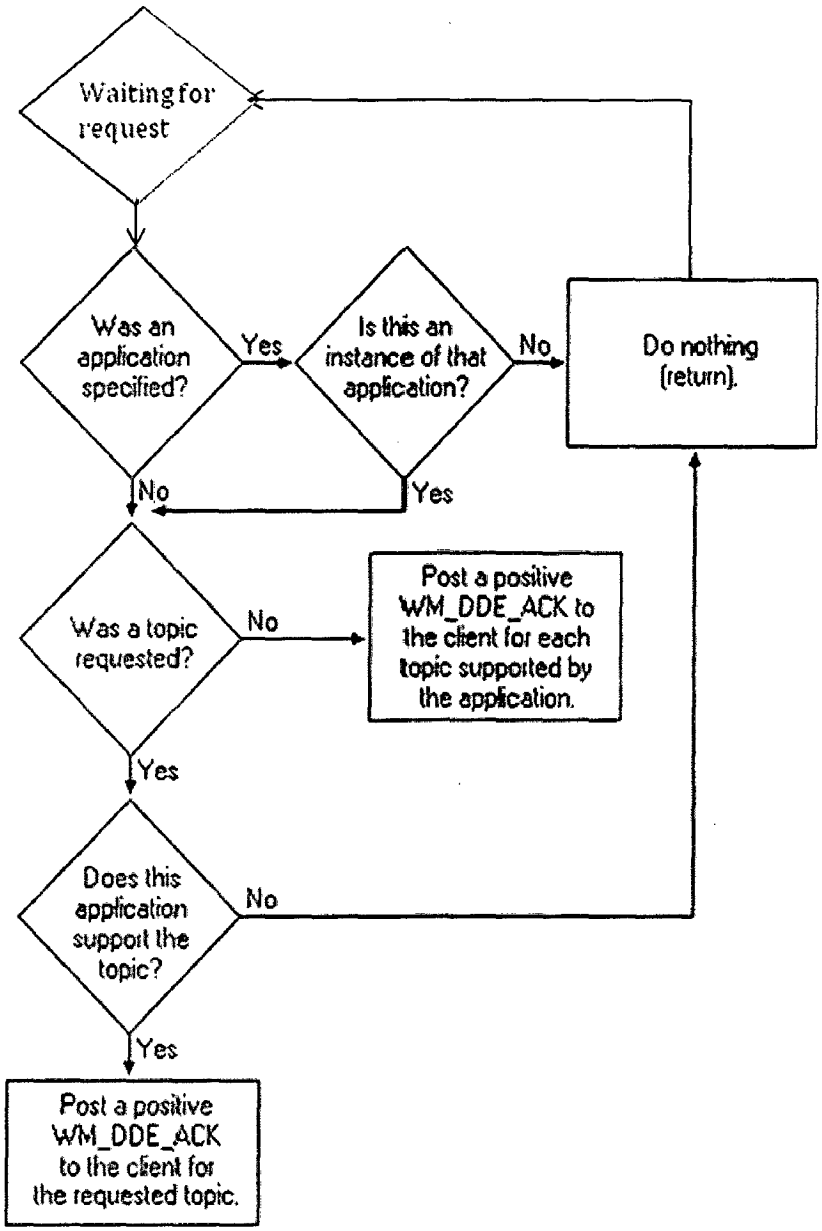


Fig 2.3 Flow chart of server response [33]

2.2.3 NET DDE [1][2][33]

The server application and the client application should be operated in the same computer in the standard DDE dialogue and it cannot be operated on the network, which is not appropriate for the integration of multi-protocol buses. Therefore NetDDE has been used

to solve this problem. NetDDE, which only adds something to the DDE is not a new data exchange format. Network DDE is a technology that allows applications that use the DDE transport to transparently exchange data over a network. Its principle is to conduct network mapping through the inner function blocks operated by the operation system in the background and map the data required by the DDE dialogue on one computer to other workstations on the network. So, applications on different computers can communicate with each other. In this way, it is possible that they can share information on the same network and the application between client-server will come true. Figure2.4 shows its communication principle.

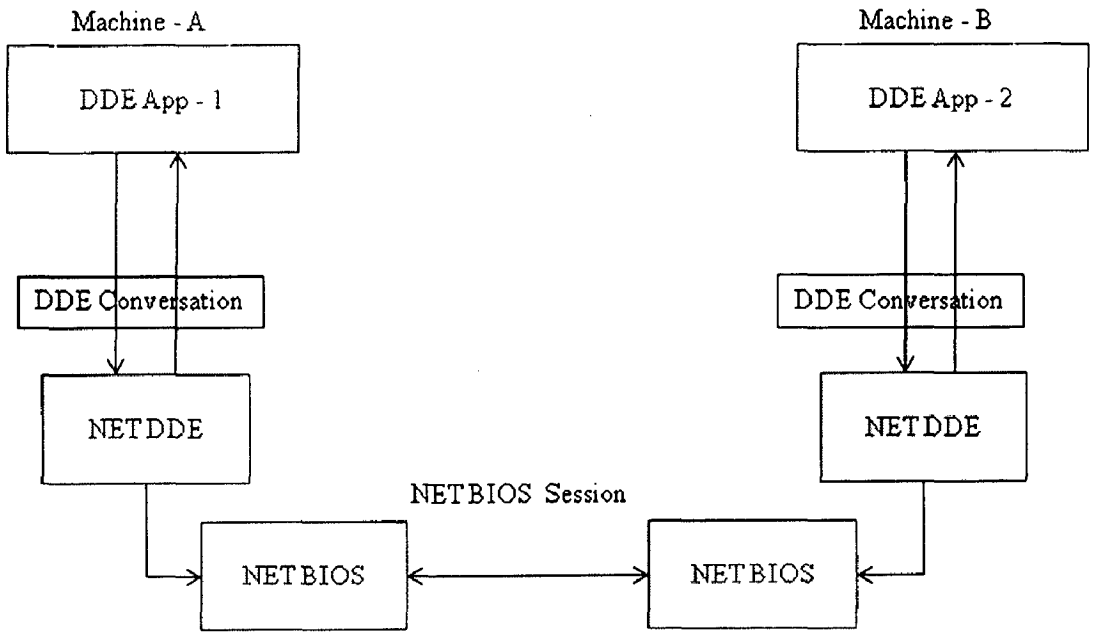


Fig 2.4 Principle of NET DDE communication

It consists of two major components:

(i) The NetDDE agent. This is a service that acts as a proxy for the remote DDE application. It communicates with all local DDE applications, and with remote NetDDE agents using NetBIOS as shown in Fig 2.4.

(ii) A DLL that implements NetDDE Application Program Interface functions such as NDdeShareAdd, NDdeShareDel, and so on. This DLL is usually named NDDEAPI.DLL.

NETDDE expose the NetBIOS interface and map NetBIOS commands to their own native commands. The NetBIOS Frames protocol (NBFP) can be implemented by the underlying protocol software to perform the network I/O required by the NetBIOS interface.

2.3 Integration based on OPC [1][2][5][6]

OPC stands for OLE for Process Control. OPC specification uses Microsoft's COM and DCOM technology for applications to exchange data on one or more computers using client-server architecture. OPC defines a set of standard objects, interfaces and methods for use in process control applications to facilitate true interoperability. OPC standardizes on technology rather than on product. By using the OPC set of standards, data can be passed from a data source to any OPC compliant application. OPC was designed to bridge Windows based applications and process control hardware and software applications. It is an open standard that permits a consistent method of accessing data from field devices. This method remains the same regardless of the type and source of data. OPC servers provide a method for many different software packages to access data from a process control device, such as a PLC or DCS. Traditionally, any time a package needed access to data from a device, a custom interface, or driver, had to be written. The purpose of OPC is to define a common interface that is written once and then reused by any SCADA, HMI, or custom software applications. The architecture of the OPC based network is shown in Fig 2.5.

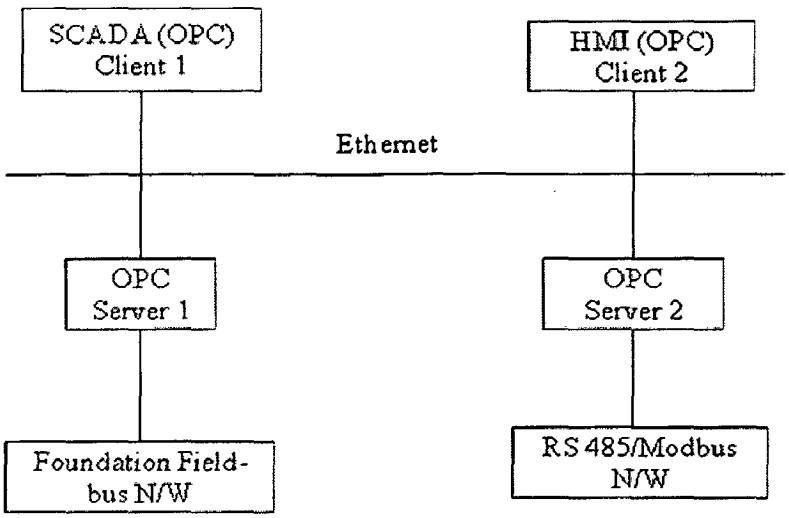


Fig 2.6 Architecture of OPC Based Network

OPC standard specifies that a manufacturer of the hardware must develop the OPC server with the standard interfaces as defined in the standard. The OPC server must poll the device for real-time data. Any client application that wants to access the data must use these standard server interfaces to access the server for real-time data. OPC servers use Microsoft's OLE (Object Linking and Embedding) technology (also known as the Component Object Model, or COM) to communicate with clients. COM technology permits a standard for real-time information exchange between software applications and process hardware to be defined. Since there are many kinds of device level networks exists the solving method of system integration with the software can adopt middle objects between system integration and different control networks. As in figure 2.6 every network provides an OPC server and client applications can call these OPC servers and get data from different buses with the unified OPC interface. The advantage is that if one kind of protocol version is updated, the relevant server application should only be added, while the other OPC servers and clients need not be modified. Integration of fieldbus system based on OPC is system level integration.

2.4 Selection for SCADA

From the schemes discussed in the above sections, the gateway based scheme is not an optimum solution as there exits many device level networks with different structures and features giving rise to the requirement of different kinds of gateway products to connect the device network to the SCADA network. If the number of device networks is large in number, the system will become bigger, as well as the price, which will make the gateway become the bottleneck of the whole system. However there are other problems to be solved to use Ethernet as the industrial network, most important is the real-time of the Ethernet. The Ethernet is a non-confirmed network based on collision detection and avoidance, so when the network flux is added the responding time will be slow. Hence gateway scheme cannot become a solution for Mixed-mode network.

The other two schemes, which are the ways Windows, allow data transfer between applications, the Dynamic Data Exchange (DDE) protocol and Object Linking and

Embedding for Process Control (OPC). The DDE protocol is a set of messages that uses shared memory to exchange data between applications. Network Dynamic Data Exchange (NetDDE) is an extension of the DDE protocol that has existed since the very early versions of Windows. NetDDE extends all of the DDE capabilities and enables applications on two or more workstations to dynamically share information over a network. NetDDE is not a special form of DDE but rather a service that examines the information contained in a DDE conversation and looks for a special application name. NetDDE Share must be configured before the applications can exchange data. Due to the limited security model for NetDDE, support for this has been waning. Although it is available on Windows XP and 2003 Server, it is disabled by default.

OPC is based on the COM/DCOM. COM is a binary standard that enables objects to interoperate in a networked environment and includes ActiveX Controls, Automation, and object linking and embedding (OLE) technologies. Since the release of Windows NT, Microsoft recommended DCOM as the preferred method for data exchange between client/server applications over the network. DCOM builds upon the remote procedure call (RPC) technology, and was designed to give developers more control over security, as compared to other interprocess communication mechanisms. From an implementer's point of view, the following points are what should be considered:[1][30]

1. **Supportability:** DDE is old and/or obsolete. Although DDE has made the cut to Windows Vista, the fact that there is no support for remote access via NetDDE will be a severe backdrop for DDE applications in the future.

2. **Interoperability:** It would seem to be an obvious point, but what applications will be connecting to your system? Are there more DDE clients or OPC clients available in the marketplace, and which are more likely to be developed in the future? Consider also that OPC offers standardized compliance testing for servers, and OPC Interoperability testing for client applications.

3. **Speed and Through-put:** One of the main differences between DDE and COM applications, OPC, is how transactions are identified and processed. Unlike COM which

uses pointers to shared memory objects, DDE applications create and exchange string and data handles, which identify strings and memory objects. Also the NetDDE service examines each DDE request, looking for the use of a special reserved application name, which is preceded by the name of the remote system. The resulting string parsing and management results in slower performance. COM is a truly binary standard, that is designed to process and move more data, more quickly

4. **Scalability:** Related to speed is the concept of scalability. As the number of client connections or item requests goes up, how is performance affected? When handling messages from more than one source, a DDE client or server must process the messages of a conversation synchronously. On the other hand COM/OPC supports the concept of non-blocking method calls. OPC Clients can exploit parallelism without the pain of multi-threading, and servers can handle calls asynchronously for vastly improved scalability. Another key feature of OPC, is that the OPC Server will cache the most current value, so only the values which have changed need to be reported back to the client application. DDE updates all values, regardless whether or not they have changed.

5) **Robustness:** In addition to the speed and scalability issues, a good error reporting and handling infrastructure is needed for a truly robust application. In addition to standard COM error handling codes, the OPC specifications outline specific error codes to be returned under different scenarios.

6) **Security:** Security in any networked architecture is another whole topic of discussion, but we can still hit the high points between DDE and OPC. Network DDE uses trusted shares and security descriptors to control access to shares. The user that created the NetDDE share must be logged on in order for the connection to occur. OPC Security is controlled by Windows DCOM security configuration. Whenever an OPC client calls a method, DCOM obtains the client's current username, and passes it to the machine where the server is running. DCOM on the server's machine then validates the username using the configured authentication mechanism and checks the access control list (ACL) for the component.

DCOM security is at the same time one of the strongest features and biggest implementation headaches of OPC, but it works.

DDE probably still has its uses for simple, small-scale, non-networked applications. However, the overwhelming majority of industrial applications will be non-trivial, large scale and involves multiple connections over the network. Finally here in deploying a mixed-mode network OPC can be preferred to DDE and other schemes as the best scheme for implantation.

Chapter – 3 OPC: Features, Technologies and Standards

3.1 OPC Features

Today's manufacturing enterprises are looking at advanced distributed object technologies as a means of integrating the islands of automation that currently exist within their worldwide operations. Distributed object technologies allow the seamless exchange of information across plant and enterprise networks. Furthermore, OLE for Process Control (OPC), work to significantly reduce the time, cost, and effort required to write custom interfaces for the hundreds of different intelligent devices and networks in use today. OPC makes it possible to have plug-and-play software and hardware across the spectrum of vendors, devices, software, and systems that manufacturing can easily integrate into corporate-wide automation and business systems.

3.1.1 Purpose of OPC [3][4][5][33]

Before OPC all the client applications that have been developed that require data from a data source and access that data by independently developing "Drivers" for their own packages. The following figure 3.1 shows diagrammatically the scenario of various client applications accessing the devices by programming the drivers for individual networks. This lead to the following problems:

Much duplication of effort: Everyone must write a driver for a particular vendor's hardware. Each application must write drivers for all the hardware it needs to access.

Inconsistencies between vendor's drivers: Hardware features not supported by all driver developers.

Support for hardware feature changes: A change in the hardware's capabilities may break some drivers.

Access Conflicts: Two packages generally cannot access the same device simultaneously since they each contain independent Drivers.

Burden on Hardware: As two applications cannot access devices simultaneously, the device must process the commands from each application in a queue this increases burden on the hardware. This leads to reduction in life time of the hardware as well.

Hardware manufacturers attempt to resolve these problems by developing drivers, but are hindered by differences in client protocols. Finally the choice application will be on the basis of how many devices it supports for.

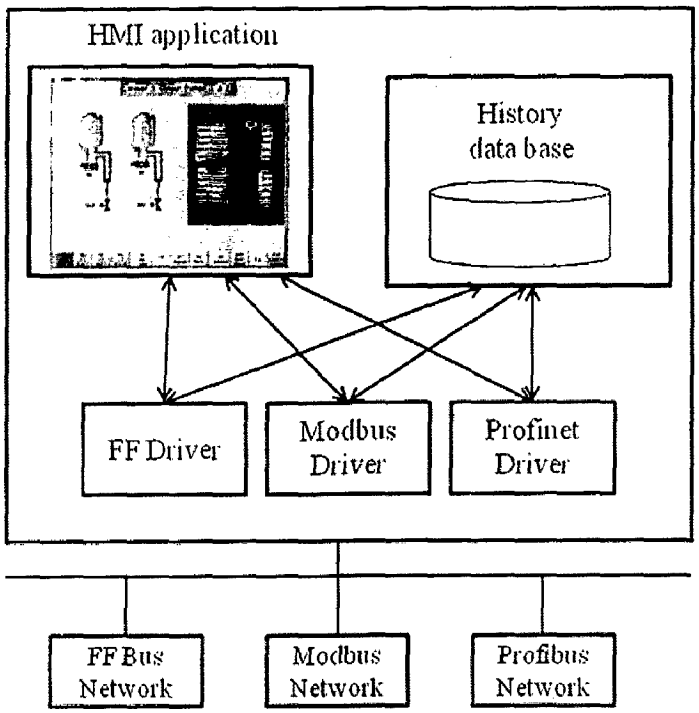


Fig 3.1 Applications without OPC interface

Today they cannot develop an efficient driver that can be used by all clients. OPC is an industry standard setup by the OPC Foundation specifying the software interface to a server that collects data produced by field devices. OPC draws a line between hardware providers and software developers. It provides a mechanism to provide data from a data source and communicate the data to any client application in a standard way. The following figure 3.2 shows a network with OPC interface. A vendor can now develop a reusable, highly optimized server to communicate to the data Source, and maintain the mechanism to access data from the data source or device efficiently. Providing the server with an OPC interface allows any client to access their devices

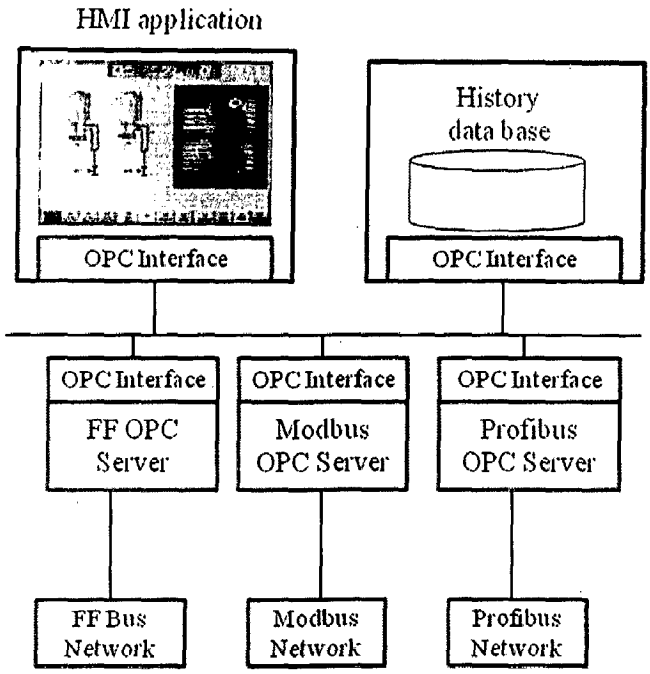


Fig 3.2 Applications with OPC interface

.3.2 OPC Technologies

The OPC Specifications resulted from the collaboration of a number of leading worldwide automation suppliers working in cooperation with Microsoft. The specifications are Originally based on Microsoft's OLE COM (component object model) and DCOM (distributed component object model) technologies, the specification defined a standard set of objects, interfaces and methods for use in process control and manufacturing automation applications to facilitate interoperability. The COM/DCOM technologies provided the framework for software products to be developed. There are now hundreds of OPC Data Access servers and clients. The OLE, COM and DCOM technologies are described in detail below before the standards are introduced.

3.2.1 Object Linking and Embedding [33][34][35]

OLE is a technology that enables an application to create compound documents that contain information from a number of different sources. OLE documents, historically called compound documents, seamlessly integrate various types of data, or components. Sound clips, spreadsheets, and bitmaps are typical examples of components found in OLE

documents. Supporting OLE in your application allows your users to use OLE documents without worrying about switching between the different applications; OLE does the switching for you. You use a container application to create compound documents and a server application or component application to create the items within the container document. Any application you write can be a container, a server, or both. OLE incorporates many different concepts that all work toward the goal of seamless interaction between applications. These areas include the following:

(a) Linking and Embedding

Linking and embedding are the two methods for storing items created inside an OLE document that were created in another application. Using the Paste command in a container application can create an embedded component, or embedded item. The source data for an embedded item is stored as part of the OLE document that contains it. In this way, a document file for a word processor document can contain text and also can contain bitmaps, graphs, formulas, or any other type of data. OLE provides another way to incorporate data from another application: creating a linked component, or linked item, or a link. The steps for creating a linked item are similar to those for creating an embedded item. Unlike an embedded component, a linked component stores a path to the original data, which is often in a separate file. Every OLE item, whether embedded or linked, has a type associated with it based on the application that created it

(b) OLE Containers and servers

A container application is an application that can incorporate embedded or linked items into its own documents. The documents managed by a container application must be able to store and display OLE document components as well as the data created by the application itself. A container application must also allow users to insert new items or edit existing items by activating server applications when necessary. A server application or component application is an application that can create OLE document components for use by container applications. Server applications usually support drag and drop or copying their data to the Clipboard so that a container application can insert the data as an embedded or

linked item. An application can be both a container and a server. Most servers are stand-alone applications or full servers; they can either be run as stand-alone applications or can be launched by a container application. Containers and servers do not communicate directly. Instead, they communicate through the OLE system dynamic-link libraries (DLL). These DLLs provide functions those containers and servers call, and the containers and servers provide callback functions that the DLLs call. Using this means of communication, a container does not need to know the implementation details of the server application. It allows a container to accept items created by any server without having to define the types of servers with which it can work. As a result, the user of a container application can take advantage of future applications and data formats. If these new applications are OLE components, then a compound document will be able to incorporate items created by those applications. Client items are data items belonging to another application that are either contained in or referenced by an OLE container application's document. Client items whose data is contained within the document are embedded; those whose data is stored in another location referenced by the container document are linked.

3.2.2 Component Object Model [33][34][35]

(a) Basic Overview

The Microsoft Component Object Model (COM) is a platform-independent, distributed, object-oriented system for creating binary software components that can interact. COM is the foundation technology for Microsoft's OLE (compound documents), ActiveX (Internet-enabled components), as well as others. To understand COM and therefore all COM-based technologies, it is crucial to understand that it is not an object-oriented language but a standard. Nor does COM specify how an application should be structured; language, structure, and implementation details are left to the application programmer. Rather, COM specifies an object model and programming requirements that enable COM objects also called COM components, or sometimes simply objects to interact with other objects. These objects can be within a single process, in other processes, and can even be on remote machines. They can have been written in other languages, and they may be structurally quite

dissimilar, which is why COM is referred to as a binary standard—a standard that applies after a program has been translated to binary machine code. The only language requirement for COM is that code is generated in a language that can create structures of pointers and, either explicitly or implicitly, calls functions through pointers. COM defines the essential nature of a COM object. In general, a software object is made up of a set of data and the functions that manipulate the data. A COM object is one in which access to an object's data is achieved exclusively through one or more sets of related functions. These function sets are called interfaces, and the functions of an interface are called methods. Further, COM requires that the only way to gain access to the methods of an interface is through a pointer to the interface.

Besides specifying the basic binary object standard, COM defines certain basic interfaces that provide functions common to all COM-based technologies, and it provides a small number of API functions that all components require. COM also defines how objects work together over a distributed environment and has added security features to help provide system and component integrity.

(b) COM Clients and Servers

A critical aspect of COM is how clients and servers interact. A COM client is whatever code or object gets a pointer to a COM server and uses its services by calling the methods of its interfaces. A COM server is any object that provides services to clients; these services are in the form of COM interface implementations that can be called by any client that is able to get a pointer to one of the interfaces on the server object. There are two main types of servers, in-process and out-of-process. In-process servers are implemented in a dynamic linked library (DLL), and out-of-process servers are implemented in an executable file (EXE). Out-of-process servers can reside either on the local machine or on a remote machine. In addition, COM provides a mechanism that allows an in-process server (a DLL) to run in a surrogate EXE process to gain the advantage of being able to run the process on a remote machine. The COM programming model and constructs have now been extended so that COM clients and servers can work together across the network, not just within a given

machine. This enables existing applications to interact with new applications and with each other across networks with proper administration, and new applications can be written to take advantage of networking features. COM client applications do not need to be aware of how server objects are packaged, whether they are packaged as in-process objects (in DLLs) or as local or remote objects (in EXEs). Distributed COM further allows objects to be packaged as Microsoft Windows NT or Microsoft Windows 2000 Services, synchronizing COM with the rich administrative and system-integration capabilities. COM is designed to make it possible to add the support for location transparency that extends across a network. It allows applications written for single machines to run across a network and provides features that extend these capabilities and add to the security necessary in a network.

(c) COM Server Responsibilities

One of the most important ways for a client to get a pointer to an object is for the client to ask that a server be launched and that an instance of the object provided by the server be created and activated. It is the responsibility of the server to ensure that this happens properly. There are several important parts to this. The server must implement code for a class object through an implementation of either the IClassFactory or IClassFactory2 interface. The server must register its CLSID in the system registry on the machine on which it resides and further, has the option of publishing its machine location to other systems on a network to allow clients to call it without requiring the client to know the server's location. The server is primarily responsible for security—that is, for the most part, the server determines whether it will provide a pointer to one of its objects to a client. In-process servers should implement and export certain functions that allow the client process to instantiate them.

(d) Registering COM Servers

After you have defined a class in code and assigned it a CLSID, you need to put information in the registry that will allow COM, on request of a client with the CLSID, to create instances of its objects. This information tells the system, for a given CLSID, where the DLL or EXE code for that class is located and how it is to be launched. There is more

than one way of registering a class in the registry. In addition, there are other ways of "registering" a class with the system when it is running, so that the system is aware that a running object is currently in the system

(e) Inter-Object Communication

COM is designed to allow clients to communicate transparently with objects, regardless of where those objects are running—in the same process, on the same machine, or on a different machine. This provides a single programming model for all types of objects, and for both object clients and object servers. From a client's point of view, all objects are accessed through interface pointers. A pointer must be in-process. In fact, any call to an interface function always reaches some piece of in-process code first. If the object is in-process, the call reaches it directly, with no intervening system-infrastructure code. If the object is out-of-process, the call first reaches what is called a "proxy" object provided either by COM or by the object. The proxy packages call parameters (including any interface pointers) and generate the appropriate remote procedure call (or other communication mechanism in the case of custom generated proxies) to the other process or the other machine where the object implementation is located. This process of packaging pointers for transmission across process boundaries is called marshaling. From a server's point of view, all calls to an object's interface functions are made through a pointer to that interface. Again, a pointer has context only in a single process, and the caller must always be some piece of in-process code. If the object is in-process, the caller is the client itself. Otherwise, the caller is a "stub" object provided either by COM or by the object itself. The stub receives the remote procedure call (or other communication mechanism in the case of custom generated proxies) from the "proxy" in the client process, unmarshals the parameters, and calls the appropriate interface on the server object. From the points of view of both clients and servers, they always communicate directly with some other in-process code. COM provides an implementation of marshaling, referred to as standard marshaling. This implementation works very well for most objects and greatly reduces programming requirements, making the marshaling process effectively transparent.

The clear separation of interface from implementation of COM's process transparency can, however, get in the way in some situations. The design of an interface that focuses on its function from the client's point of view can sometimes lead to design decisions that conflict with efficient implementation of that interface across a network. In cases like this, what is needed is not pure process transparency but "process transparency, unless you need to care." COM provides this capability by allowing an object implementer to support custom marshaling. Standard marshaling is, in fact, an instance of custom marshaling—it is the default implementation used when an object does not require custom marshaling. You can implement custom marshaling to allow an object to take different actions when used from across a network than it takes under local access—and it is completely transparent to the client. This architecture makes it possible to design client/object interfaces without regard to network performance issues and then later to address network performance issues without disrupting the established design. COM does not specify how components are structured; it specifies how they interact. COM leaves the concern about the internal structure of a component to programming languages and development environments. Conversely, programming environments have no set standards for working with objects outside of the immediate application. Microsoft Visual C++, for example, works extremely well for manipulating objects inside an application but has no support for working with objects outside the application. Generally, all other programming languages are the same in this regard. Therefore, to provide network wide interoperability, COM, through language-independent interfaces, picks up where programming languages leave off. The double indirection of the structure means that the pointers in the table of function pointers do not need to point directly to the real implementation in the real object. This is the heart of process transparency.

For in-process servers, where the object is loaded directly into the client process, the function pointers in the table point directly to the actual implementation. In this case, a function call from the client to an interface method directly transfers execution control to the method. However, this cannot work for local, let alone remote, objects because pointers to memory cannot be shared between processes. Nevertheless, the client must be able to call interface methods as if it were calling the actual implementation. Thus, the client uniformly

transfers control to a method in some object by making the call. A client always calls interface methods in some in-process object. If the actual object is local or remote, the call is made to a proxy object, which then makes a remote procedure call to the actual object. So what method is actually executed? The answer is that whenever there is a call to an out-of-process interface, each interface method is implemented by a proxy object. The proxy object is always an in-process object that acts on behalf of the object being called. This proxy object knows that the actual object is running in a local or remote server. The proxy object packages up the function parameters in some data packets and generates an RPC call to the local or remote object. That packet is picked up by a stub object in the server's process on the local or a remote machine, which unpacks the parameters and makes the call to the real implementation of the method. When that function returns, the stub packages up any out-parameters and the return value and sends it back to the proxy, which unpacks them and returns them to the original client. Thus, client and server always talk to each other as if everything was in-process. All calls from the client and all calls to the server are, at some point, in-process. But because the vtbl structure allows some agent, like COM, to intercept all function calls and all returns from functions, that agent can redirect those calls to an RPC call as necessary. Although in-process calls are faster than out-of-process calls, the process differences are completely transparent to the client and server.

3.2.3 Distributed Component Object Model (DCOM) [33][34][35]

The Microsoft Distributed Component Object Model (DCOM) extends the Component Object Model (COM) to support communication among objects on different computers—on a local area network (LAN), a wide area network (WAN), or even the Internet. With DCOM, your application can be distributed at locations that make the most sense to your customer and to the application.

(a) The DCOM Architecture

DCOM is an extension of the Component Object Model (COM). COM defines how components and their clients interact. This interaction is defined such that the client and the component can connect without the need of any intermediary system component. The client

calls methods in the component of a server without any overhead whatsoever. Figure 3.3 illustrates this in the notation of the Component Object Model.

Fig 3.3 COM components in the same process [33]

In today's operating systems, processes are shielded from each other. A client that needs to communicate with a component in another process cannot call the component directly, but has to use some form of interprocess communication provided by the operating system. COM provides this communication in a completely transparent fashion. It intercepts calls from the client and forwards them to the component in another process. Figure 3.4 illustrates how the COM/DCOM run-time libraries provide the link between client and the server components running in different processes.

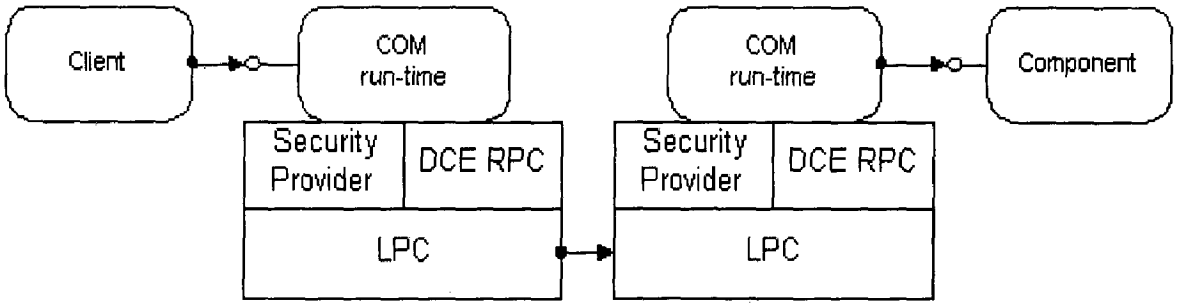


Fig 3.4 Client and Server in different processes[33]

When client and component reside on different machines, DCOM simply replaces the local interprocess communication with a network protocol. Neither the client nor the component is aware that the wire that connects them has just become a little longer. Figure 3.5 shows the overall DCOM architecture. The COM run-time provides object-oriented services to clients and server components and uses RPC and the security provider to generate standard network packets that conform to the DCOM wire-protocol standard.

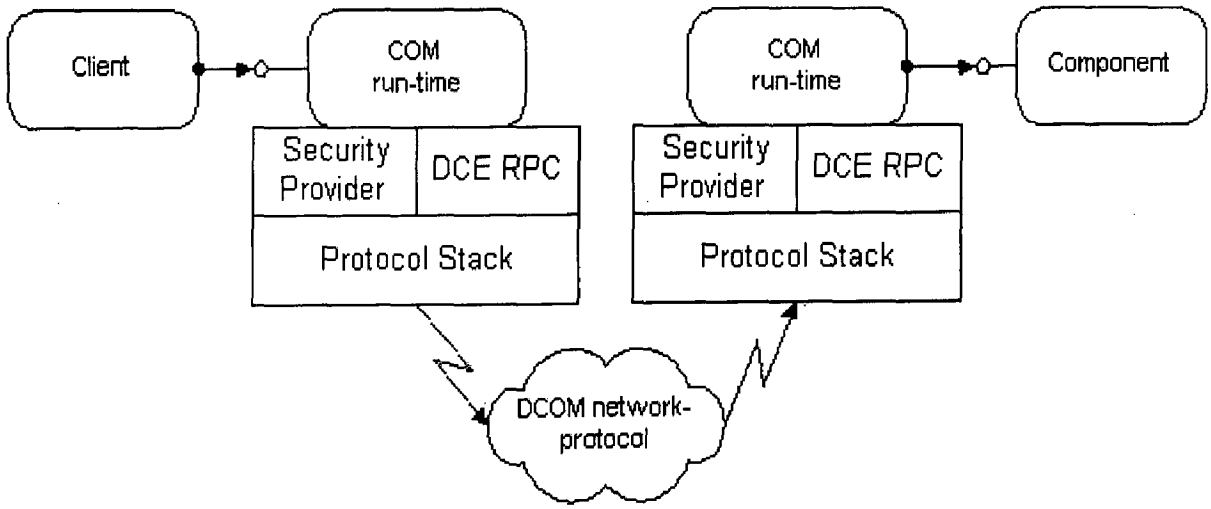


Fig 3.5 Client and Server on different machines[33]

While COM is a specification for building interoperable components, Distributed COM (DCOM) is simply a high-level network protocol designed to enable COM-based components to interoperate across a network. We consider DCOM a high-level network protocol because it is built on top of several layers of existing protocols. It may also be useful to think of a DCOM protocol stack in terms of the Open Systems Interconnection (OSI) seven-layer model. In Figure 3.6 the OSI seven-layer model is juxtaposed with the sample protocol stack discussed here. Note that this figure shows a Windows platform. Other operating systems might implement the protocols at different layers.

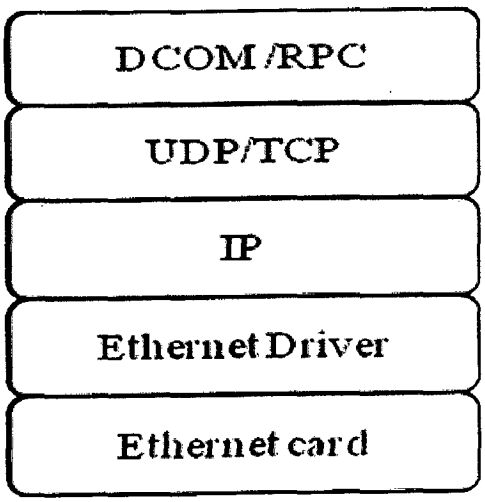


Fig 3.6 DCOM protocol stack [35]

3.2.4 ActiveX Technology [33]

ActiveX is an open integration platform that provides developers, users, and Web producers a fast and easy way to create integrated programs and content for the Internet and Intranets. ActiveX is a standard that enables software components to interact with one another in a networked environment, regardless of the language(s) used to create them. Most World Wide Web (WWW) users will experience ActiveX technology in the form of ActiveX controls, ActiveX documents, and ActiveX scripts. ActiveX controls, formerly known as OLE controls or OCX controls, are components (or objects) that you can insert into a Web page or other program so that you can reuse packaged functionality that someone else programmed. When you are browsing with an ActiveX-aware Web browser, such as Internet Explorer, ActiveX documents enable you to open a program with its own toolbars and menus available. This means you can open non-HTML files, such as Microsoft Excel or Microsoft Word files, by using an ActiveX-aware Web browser.

3.2.5 OPC Server Structure and Functionality [36]

According to OPC standards every manufacturer of the devices has to develop their own server complaint to OPC standards, which describes the functions of the server. All the functions are ordered one over the other as in Fig 3.6.

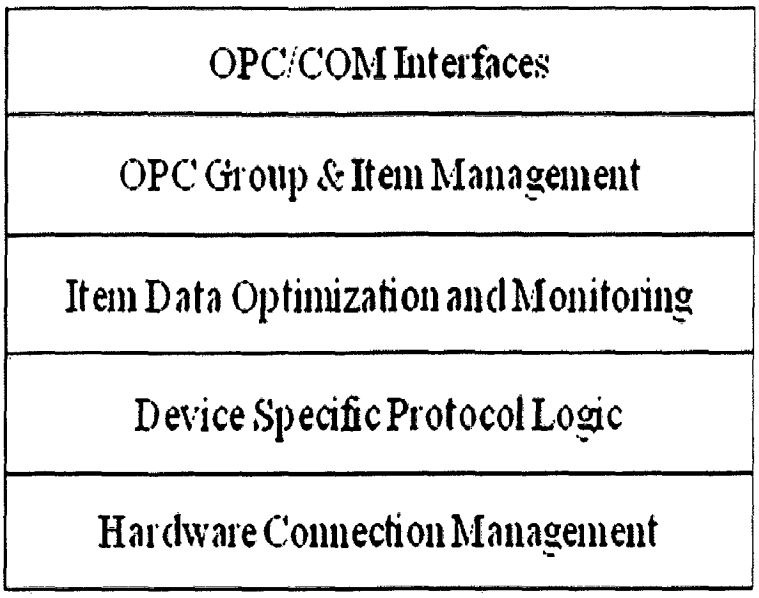


Fig 3.7 Functional Block Diagram of an OPC Server

(a) OPC / COM Interfaces [7][8][9][36]

This specification describes the OPC COM Objects and their interfaces implemented by OPC Servers. The specification specifies about what the interfaces must be, but does not talk about how of the implementation of these interfaces. OPC specifications always contain two sets of interfaces; Custom Interfaces and Automation interfaces. As shown in Figure 3.8

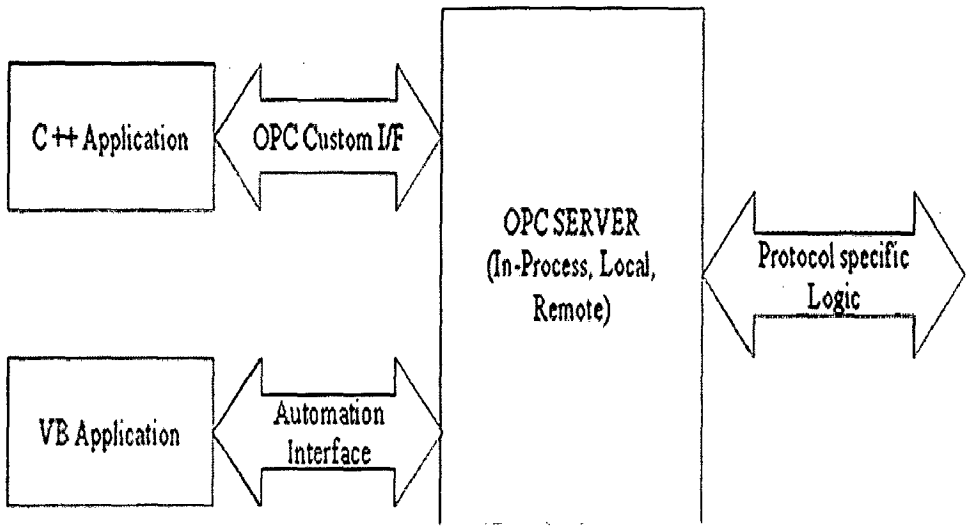


Fig 3.8 OPC Interfaces

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. OPC server developers must implement all functionality of required interfaces OPC server developers may implement the functionality of the optional interfaces An optional interface is one that the server developer may elect to implement. In general, client programs which are created using scripting languages will use the automation interface. Client programs which are created in C++ will find it easiest to use the custom interface for maximum performance.

(b) OPC Group &Item Management

An OPC server is structured as a directory with root, branches and leaves. The branches may consist of sub branches and items. The branches are called groups. A group consists of related data items. The scenario is well illustrated in fig 3.9.

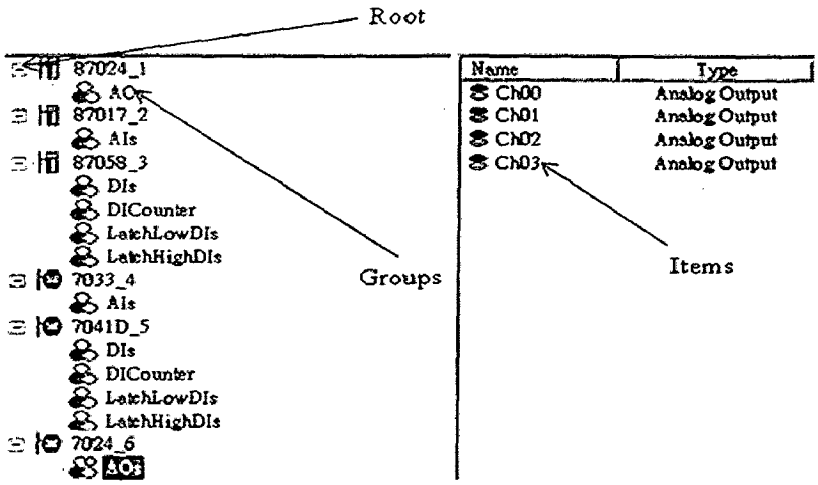


Fig 3.9 OPC server structure

The structure may be flat or Hierarchical model. The groups and items are defined in the server during the configuration of the network. Every item is identified by its fully qualified item ID. For example the fully qualified item ID of an item indicated in fig 3.9 will be "87024_1.AO.Ch03".

(c) Item Data Optimization and Monitoring

The OPC standards just define what the interfaces are but do not define their implementation. Hence the implementer of server should develop the logic for the interface such that it should optimize the inputs and outputs with the hardware. The logic should be such that it should reduce the unnecessary burden on the hardware by continuously polling the devices, rather it should fire the events when data changes or any event occurs and update the clients continuously through cache.

(d) Device specific protocol logic & Hardware Connection management

It is the function of the server to communicate with devices in their respective protocol. That is formatting the data into respective protocol formats and it also takes care of the hardware connection management like opening the port closing the port etc.

3.2.6 Client – Server Communications [34][36]

The communication between OPC client and OPC server depends on location of the client and the server. The situations can be of three types, in-process, same machine and on the remote machine. The following fig 3.10 and 3.11 illustrate the way the client and server

communicate while residing on the same machine and remote machine respectively. The Clients and servers residing on the same machine may be running in the same process or running in two separate processes, however the mechanism would similar. The one limitation for the DCOM is that it does not work with firewalls.

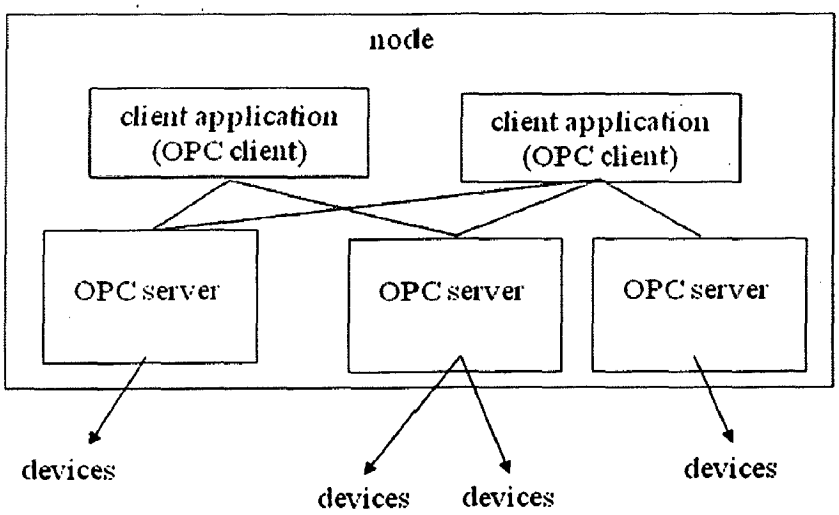


Fig 3.10 Client and Server residing on same machine

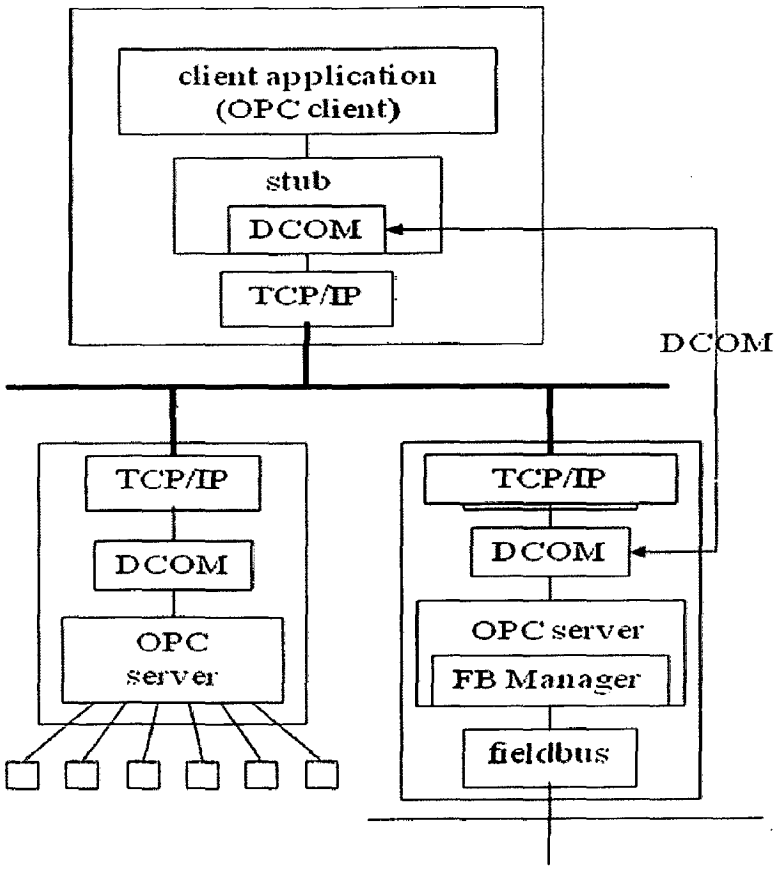


Fig 3.11 Client and Server residing on Different machines

3.2.7 OPC STANDARDS [6][7][8]

OPC is open connectivity in industrial automation and the enterprise systems that support industry. Interoperability is assured through the creation and maintenance of open standards specifications. There are currently seven standards specifications completed or in development. There are namely OPC Data Access, Alarm &Event, Historical Data Access,

(1) OPC Data Access specification

The Data Access specification defined a standard set of objects, interfaces and methods for use in process control and manufacturing automation applications to access real time values of the data items. At a high level, an OPC Data Access Server is comprised of several objects: the Server, the Group, the Item and the Browser. The OPC server object maintains information about the server and serves as a container for OPC group objects. The OPC group object maintains information about itself and provides the mechanism for containing and logically organizing OPC items. The logical relation is illustrated in figure 3.12.

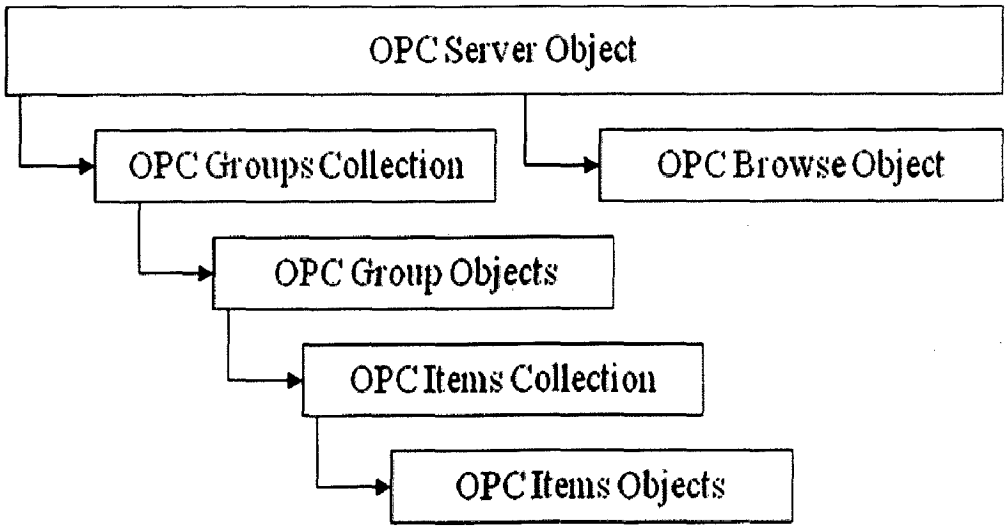


Fig 3.12 OPC Automation Server Object Model

The OPC Groups provide a way for clients to organize data. Data can be read and written. An OPC client can configure the rate that an OPC server should provide the data changes to the OPC client. OPC standard defines a set of interfaces to operate on these objects.

There are two types of groups, public and local (or 'private'). Public is for sharing across multiple clients, local is local to a client. The OPC Items represent connections to data sources within the server. 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 is 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. Note that the items are not the data sources - they are just connections to them. 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. A group can be activated and deactivated as a unit. A group also provides a way for the client to 'subscribe' to the list of items so that it can be notified when they change. Two collections were also specified Groups and Items. OLE Automation collections are objects that support Count, Item, and a hidden property called new-Enum. Any object that has these properties as part of the interface can be called a collection.

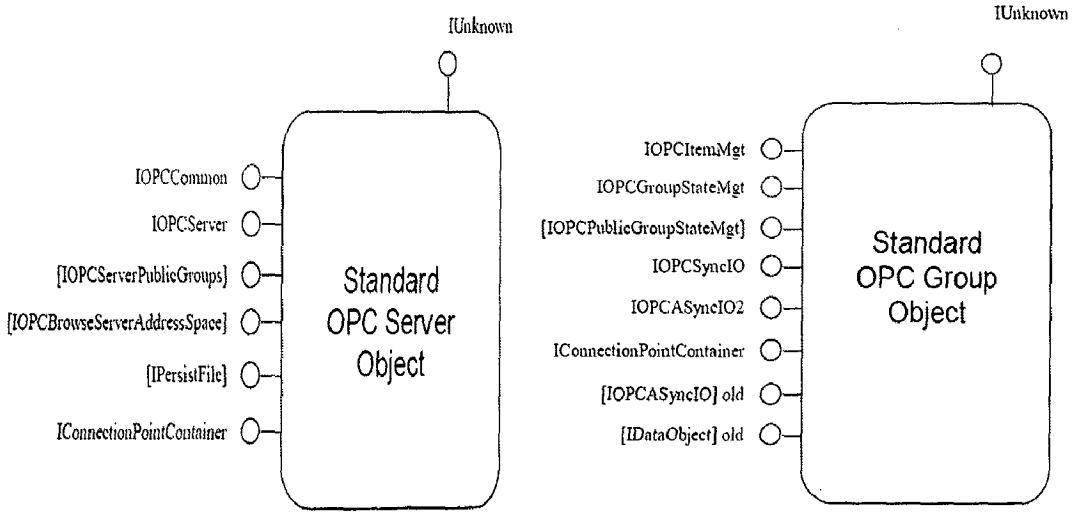


Fig 3.13 OPC Server & Group Objects with their DA interfaces

(2) OPC Alarms and Events [7][30]

The Alarm and Event specifications provide alarm and event notifications on demand in contrast to the continuous data flow of Data Access. These include process alarms, operator actions, informational messages, and tracking/auditing messages. These standards

provide the mechanisms for OPC Clients to be notified of the occurrence of specified events and alarm conditions. They also provide services which allow OPC Clients to determine the events and conditions supported by an OPC Server, and to obtain their current status. We make use of entities commonly referred to in the process control industry as *alarms* and *events*. The terms *alarm* and *event* are often used interchangeably and their meanings are not distinct. Within OPC, an *alarm* is an abnormal *condition* and is thus a special case of a *condition*. A *condition* is a named state of the OPC Event Server, or of one of its contained objects, which is of interest to its OPC Clients. On the other hand, an *event* is a detectable occurrence which is of significance to the OPC Server, the device it represents, and its OPC Clients. An event may or may not be associated with a condition. The transitions into HighAlarm and Normal conditions are events which are associated with conditions. However, operator actions, system configuration changes, and system errors are examples of events which are not related to specific conditions. OPC Clients may subscribe to be notified of the occurrence of specified events.

The alarm Server interfaces provide methods enabling the OPC Client to

- Determine the types of events which the OPC Server supports.
- Enter subscriptions to specified events, so that OPC Clients can receive notifications of their occurrences. Filters may be used to define a subset of desired events.
- Access and manipulate conditions implemented by the OPC Server.

(3) OPC Historical Data Access [7]

Historical engines today produce an added source of information that must be distributed to users and software clients that are interested in this information. Currently most historical systems use their own proprietary interfaces for dissemination of data. There is no capability to augment or use existing historical solutions with other capabilities in a plug-n-play environment. This requires the developer to recreate the same infrastructure for their products as all other vendors have had to develop independently with no interoperability with any other systems. In keeping with the desire to integrate data at all levels of business,

historical information can be considered to be another type of data. There are several types of historian servers. Some key types supported by this specification are:

• Simple Trend data servers: These servers provided little else then simple raw data storage. (Data would typically be the types of data available from an OPC Data Access server, usually provided in the form of a tuple [Time Value & Quality])

• Complex data compression and analysis servers: These servers provide data compression as well as raw data storage. They are capable of providing summary data or data analysis functions, such as average values, minimums and maximums etc. They can support data updates and history of the updates. They can support storage of annotations along with the actual historical data storage.

(4) OPC XML DA [4][28][38]

OPC has not defined a mechanism to detect nodes with OPC-XML-DA Servers or to detect OPC-XML-DA Servers on a specific node. The Universal Description Discovery and Integration (UDDI) protocol is a widely used standard for web services and it will be the likely basis for any future OPC specification for web service discovery. Until then, an OPC-XML-DA client needs to know the URL of any OPC-XML-DA server it wants to use. Note that web service implementations never need to know their URLs since the person deploying and maintaining the web service on a particular machine always assigns them. In addition, these URLs allow for an optional port number, as a result, OPC-XML-DA web services are not required to use the standard HTTP Port 80 provided the web server used supports configurable port numbers. Applications implementing or using Web Services interact with each other by using the Simple Object Access Protocol (SOAP). SOAP combines XML so as to encapsulate messages in a format suitable for transmission using HTTP Internet protocol. In short, SOAP runs within TCP/IP. TCP/IP can of course use different network proto-cols. To implement a Web Service a device has only to support HTTP and XML. Therefore OPC XML DA can be implemented on any device supporting these two properties. HTTP is firewall-friendly. This allows OPC XML to run over the internet. Defining XML schemas is pretty easy, but OPC is all about interoperability, and defining XML schemas is not enough to ensure multi-vendor compatibility. The

specification would also specify communication protocols, discovery mechanisms, error handling and all of the rest.

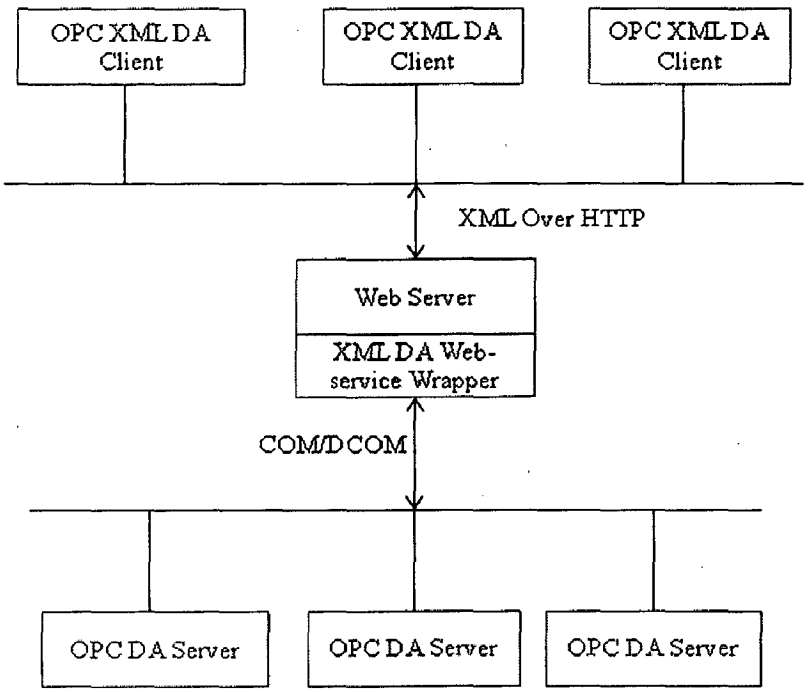


Fig 3.14 Web server Architecture

Major platform vendors like Microsoft were hard at work layering new standards and new tools on top of the existing XML standards. First, SOAP (Simple Object Access Protocol) is layered on top of XML, then WSDL (Web Services Description Language) is layered on top of SOAP. Together, these specifications define the plumbing that is needed for true interoperability.

(5) OPC Security [6][7]

All the OPC servers provide information that is valuable to the enterprise and if improperly updated, could have significant consequences to plant processes. OPC Security specifies how to control client access to these servers in order to protect this sensitive information and to guard against unauthorized modification of process parameters.

Chapter 4 Hardware and Software for SCADA System

This chapter discusses hardware and software aspects of various Field devices used for the project work.

4.1 Hardware Used

To develop a mixed-mode network Foundation fieldbus network and Modbus network were used. The hardware comprises of smart transmitters of Foundation fieldbus and data acquisition modules of Modbus protocols which were explained in below sections.

4.1.1 FF Device Descriptions

(a) Fieldbus Universal Bridge (DFI 302) [22]

The DFI302 is a powerful multifunction hardware component integral to the modular SYSTEM302 that includes the most up-to-date hardware and software necessary to manage, monitor, control, maintain and operate the plant. It is a single integrated unit with functions of interfacing, linking device, bridge, controller, gateway, Fieldbus power supply and distributed I/O subsystem. The DFI302 is total modular and has the following basic settings:

Hardware

- DF01 - Rack with 4 Slots (Backplane)
- DF02 - Last rack terminator
- DF50 - Power Supply for Backplane
- DF51 - DFI302 Processor with 1x 10 Mbps Ethernet, 1x RS-232, and 4x H1 Channels
- DF52 - Power Supply for Fieldbus
- DF53 - Power Supply Impedance for Fieldbus (4 ports)
- DF54 - Standard Ethernet Cable Twisted-Pair (100Base-TX) - Length 2 meters.

Software

- DFI OLE Server
- System302

Transmitters are employed for monitoring the field values. Various transmitters from Smar Corporation are used as a part of the project. Each transmitter has a sensor assembly, main circuit board and display board. The sensor assembly senses the input and transfers it to the main circuit board which contains CPU and memory. The display board contains the display controller and The LCD display.

A brief description of the transmitters configured is given below.

(b) Foundation Fieldbus Pressure Transmitter (LD 302) [17]

The transmitter can be used for the measurement of absolute, differential and gauge pressure, level and flow.

Principle of Operation

The transmitter has a capacitive sensor with two fixed plates corresponding to two inputs and a moving sensor diaphragm between the plates. The deflection in diaphragm is proportional to the difference of pressures applied on either sides of the diaphragm. Flow is directly proportional to the square root of differential pressure. Level transmitter has only one input while the other input is sealed.

(c) Foundation Fieldbus Position Transmitter (TP 302) [18]

The transmitter can measure the position of the valve along its calibrated range. The output is in the form of the percentage of valve opened, i.e. 0% for fully shut and 100% for fully open.

Principle of Operation

The transmitter works on the principle of Hall Effect. Which implies that when a transverse magnetic field is applied on a current carrying conductor, a potential difference appears across the conductor in a direction perpendicular to both the magnetic and electric fields?

(d) Foundation Fieldbus Temperature Transmitter (TT 302) [20]

The transmitter is mainly intended for the measurement of temperature using RTD or thermocouples, but can also accept input from other sensors with resistance or milli volt output, such as pyrometers, load cells etc.

There are four sensor terminals in order to handle both single and dual inputs.

(e) 4-20mA to Foundation Fieldbus Converter (IF 302) [19]

The IF 302 is a converter mainly intended to interface analog transmitters to a Fieldbus network. The converter can handle up to three current signal inputs both 4-20 mA and 0-20mA and makes them available to the Fieldbus system.

(f) Fieldbus Terminator (BT 302) [21]

The primary function of the bus terminator is to avoid reflection of the signal which causes major distortions on the original signal. As per the standard, the terminators shall present an impedance Z equal to 100 Ω ±2%, over the frequency range of 7.8 kHz to 39 kHz.

4.1.2 Modbus Device Descriptions

The I-7000 series modules from ICPDAS make and ND-6000 series modules of NuDam make are used for Modbus. The 7000 series is a family of RS 485 remote controllable and data acquisition modules. They provide Analog input, Analog output, Digital I/O, Timer/Counter and other functions. These modules can be controlled remotely by a set of commands. Communication between the module and the host is in ASCII format via an RS-485 bi-directional serial bus standard. Baud Rates are software programmable and transmission speeds of up to 115.2K baud can be selected.

(b) RS 232 – RS 485 Convertor

The conventional two-wire RS-485 network uses a converter module I-7520 to convert host RS-232 to a two-wire RS-485 signal and vice versa. The baud rate and data format must be set to a fixed value for the whole network. For the network of I-7000 series modules a baud rate of 9600 bits per sec and data format of 10 bits per character should be selected. The 7000 RS-485 network is the most powerful and flexible two-wire RS-485 networks in the world. It is a multiple baud rate and multiple data format network system. The 7520, RS-232 to RS-485 converter, equips a "Self Tuner" inside, therefore it can detect the baud rate and data format automatically and control the direction of the RS-485 network precisely.

(c) Analog Input and Thermocouple Modules

The analog input modules used are I-7019 R and ND-6017. ND-6011, a thermocouple input module which can sense temperature by directly connecting a thermocouple at its input terminals.

The **I-7019** module consists of eight analog input channels. Each channel should be configured separately with respect to the range of the sensor input connected. It can sense a wide range of inputs like voltage, current and temperature as well.Cold junction compensation is provide with in the module. For current input of +/- 20mA, no external resistance is required but the jumper of the corresponding channel should be shorted.

NuDAM-6017 is an analog input module with 8 input channels. Six of the eight channels are differential type and the other two are single ended type. The whole module should be configured for a single type of input. It can also sense a current input of range +/- 20mA but an external resistance of 125 ohms is to be connected and the jumper for the corresponding channel should be shorted.

NuDAM-6011/D is a multi-function analog input module with cold junction compensation (CJC). The maximum input voltage range of analog input channel is ±2.5V. The high gain feature allows very small full range of ±15mV. Temperature can be measured by directly connecting the thermocouple because of the presence of CJC inside and the high gain feature. The module provides the analog signal monitor or the alarm function. The high and low bound of the alarm limit is programmable. The alarm status can be sent to digital output channels if this function is ON. The supervisor of a factory can 'see' or 'hear' the alarm if the digital output channel control a real alarm device. The two digital output channels can be set for general-purpose use if the alarm is disabled. The module provides another one digital input channel. This can be used for general purpose such as monitor digital signal, or be used as input of the event counter.

(d) Digital Input-Output Modules

The DIO modules support TTL signal, photo-isolated digital input, relay contact output , solid-state relay output, Photo MOS output and open-collector output. The Digital I/O moduleI-7050D is used. The module has eight digital output channels and seven digital input channels.

(e) Analog Output Module

ND-6021 analog signal output module is used. It receives the digital command from host computer through RS-485 network. The format of the digital value can be engineering units, hexadecimal format or percentage of full-scale range (FSR). A microprocessor is used to convert the digital command to digital value to send to DAC. The DAC converts the digital value into analog form. The analog output can be either voltage or current output. It provides many safety functions such as isolation, watchdog, and power on safe value. The opto-isolators provide 5000Vrms isolation voltage to isolate the digital section and the remote controlled analog equipments. The damage of power surges is avoided. Another safety function is the watchdog. Whenever the host has lost contact with the remote NuDAM module, or the micro-processor is down, the module will reset itself and send the safety value to the analog output therefore the industry safety is guarantee. The safety value / power-up value can be set by configuration software.

4.2 Software Used

4.2.1 SYSCON [23]

SYSCON is the acronym for System Configuration Software. It is a part of the System 302 enterprise automation solution. Its main feature is to instantiate, configure and monitor a Foundation field bus device for probing the data collected by the field device. The software provides an explorer style of view, possess several software function blocks like analog input block, PI block, PID block etc, hence provides a way to implement complex control strategies. The function blocks are sequence of programming steps that will be executed with real-time values of the parameters as inputs. The control strategy so developed will be downloaded into the device supporting distributed control. The figure 4.1 shows a syscon window with configured devices and a simple control strategy.

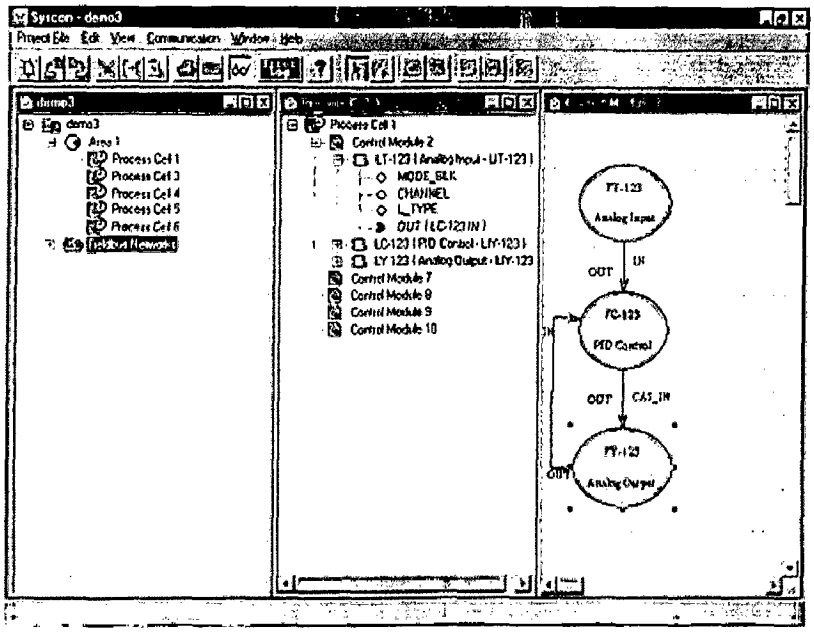


Fig 4.1 SYSCON Configuration File

4.2.2 Utility software

The Utility is a program based on COM port interface, which search modbus modules and configures the individual modules. For configuration of the network, parameters like baud rate, com port for search, timeout setting and check sum are set. The figure 4.2 shows a view of the utility software.

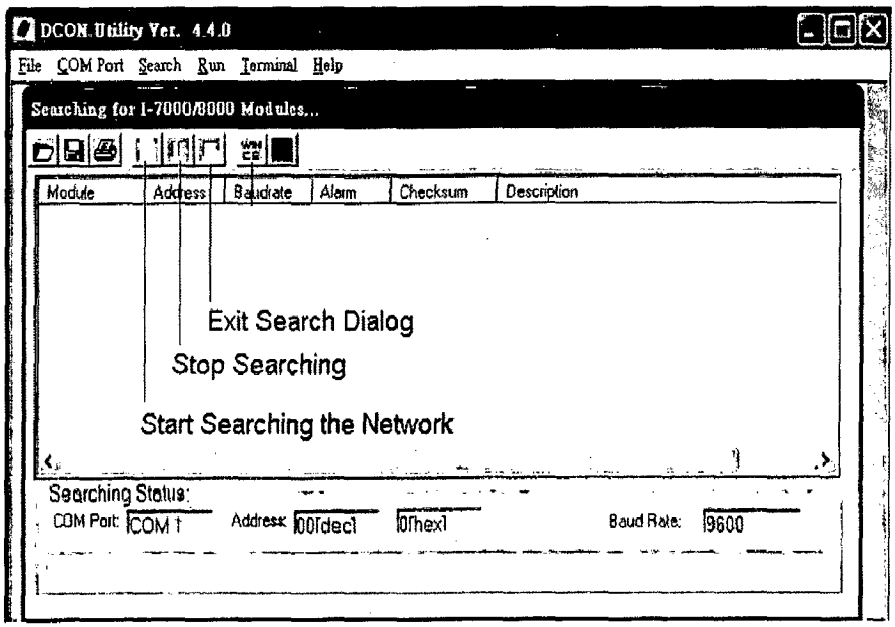


Fig 4.2 I-7000 Utility

The modules are connected to the RS 485 network in multi-drop topology and this network connects to one of the COM ports (COM 1/COM 2) of the PC through the convertor module, I-7520. Address ranging from 00 to 255 will be automatically assigned to the modules connected in the network. For configuring the I-7000 modules "%AANNTTCCFF" command should be used. For analog input modules "$AA7CiRrr" should also be used. AA – the present address of the module. NN – New address to be assigned to the module. TT – New type code. CC- Baud rate code. FF- Used to set data format. If any address conflicts occur then the conflicting address of one of the modules has to be changed using "%AANNTTCCFF" command.

4.2.3 OPC Servers

(a) Smar DFI OLE Server [14]

After fully building the system in SYSCON and checking that the devices communicate with the control station (Computer), the parameter tags of all the field devices are exported and stored in a file using the 'Export tags' option. Once the transmitters are configured they start sending the measured values continuously to the linking device and from there to the computer without any sending any further requests. The smar DFI OLE server runs continuously on the control station updating the real-time values of the tags. Smar DFI OLE server allows any OPC client to access standard Fieldbus data using the standard interface OLE for Process Control (OPC). These Servers provide also Fieldbus configuration using OLE interfaces so, all the supervision and configuration steps could be done across a network environment using Microsoft DCOM. Smar OLE Server implements the most efficient way to exchange data between the Fieldbus network and HMI. All the parameters in the Fieldbus (including the status) are available just using browse interface.

(b) NAPOPC Server [13]

The NAPOPC DA Server, the OPC server for I-700 series modules, uses an Explorer-style user interface to display a hierarchical tree of modules and groups with their associated tags as shown in figure 4.3. A group can be defined as a subdirectory containing one or more tags. A module may have many subgroups of tags. All tags belong to their module when they are scanned for perform I/O. The "OPC" stands for "OLE for Process Control" and the "DA" stands for "Data Access".

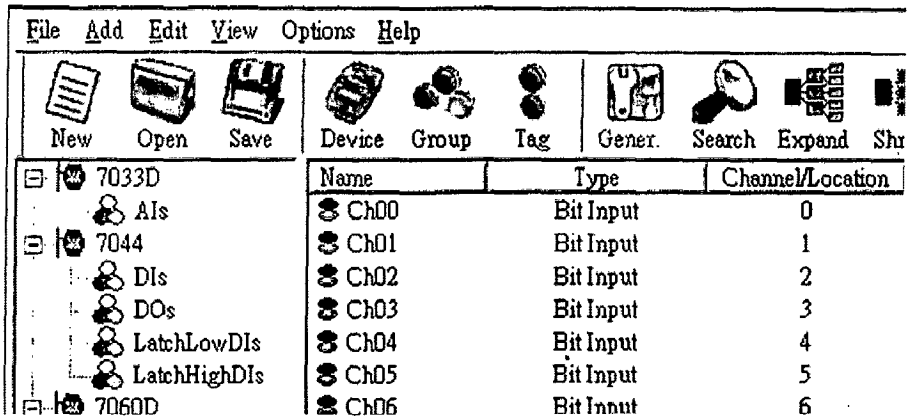


Fig 4.3 NAPOPC server window

The other functions of the NAPOPC server are:

(i) Searching the modules on the network as shown in figure 4.4. Specifies the communicate timeout value for each module. The default value is 500 (equal to 0.5 Seconds), measured in millisecond(s) [0.001 Second(s)]. After a module has been found, this timeout value will also be recorded for further use. Users can reduce this value to shorten the search time. Be careful. A shorter search time may cause communication failure

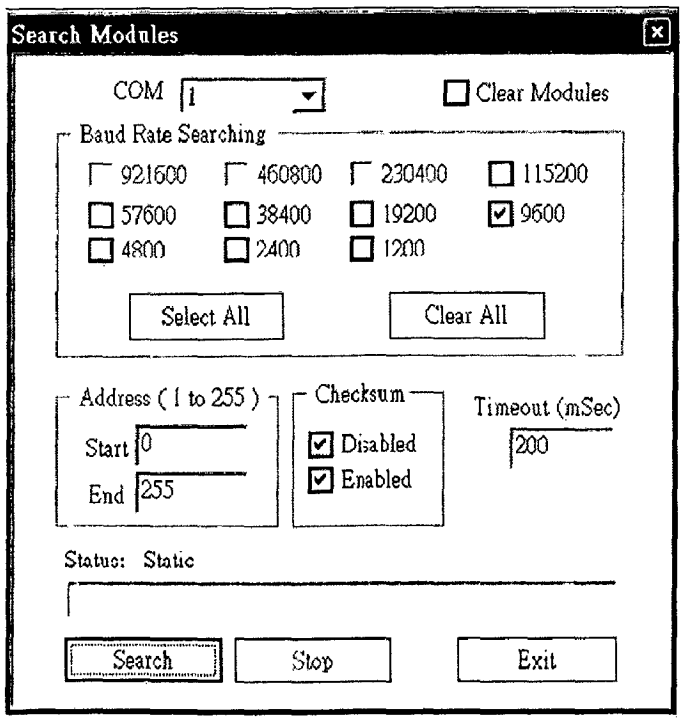


Fig 4.4 Search Window of NAP OPC server

(i) Adds devices ,groups and tags

(ii) Monitors continuously added tags

(iii)Provides the data to a number of OPC clients

(c) NuDAM OPC Server [15]

NDS-OPC includes a full-function OPC sever for NuDAM RS 485 modules, providing immediate compatibility with a very wide range of application software systems. Any software system with OPC client capabilities can access the NDS-OPC server, monitoring and control data to and from the NuDAM hardware. The NDS-OPC includes an OPC server program and an explorer program that is a graphical configuration utility for easy system setup. The explorer program saves the setup procedure during installation by integrating both hardware and software configuration into one step. Using the explorer program first, you can interactively configure NuDAM modules, setting programmable hardware parameters such as range, data format, filters. Secondly, you can configure the operation of the OPC server. You can set I/O scan times and timeout limits, as well as name each data item (tag). OPC client software can then access these data items, or tag, via this name assigned in the explorer program.

(d) Alarm &Event Server [30]

AlarmWorX is the OPC-compliant alarming software based on the OPC Alarm and Events (AE) Specification. The AlarmWorX Server receives field data from any OPC-compliant Data Access server and performs alarm detection and reporting based on the OPC Alarm and Events Standard. The event notifications generated by the AlarmWorX Server are sent to any OPC Alarm and Event clients that subscribe. The AlarmWorX Viewer and the AlarmWorX Logger are two examples of clients that can receive these notifications from the server. The architecture of Alarmworx server is well depicted in figure 4.5. The Server Application has no user interface and may optionally be run as a service on Windows. The server reads its configuration information from a Microsoft Access database file, which can be configured by the alarm server configurator application. During runtime the server polls the configuration database for changes, so configuration may be done on the fly without stopping the server.

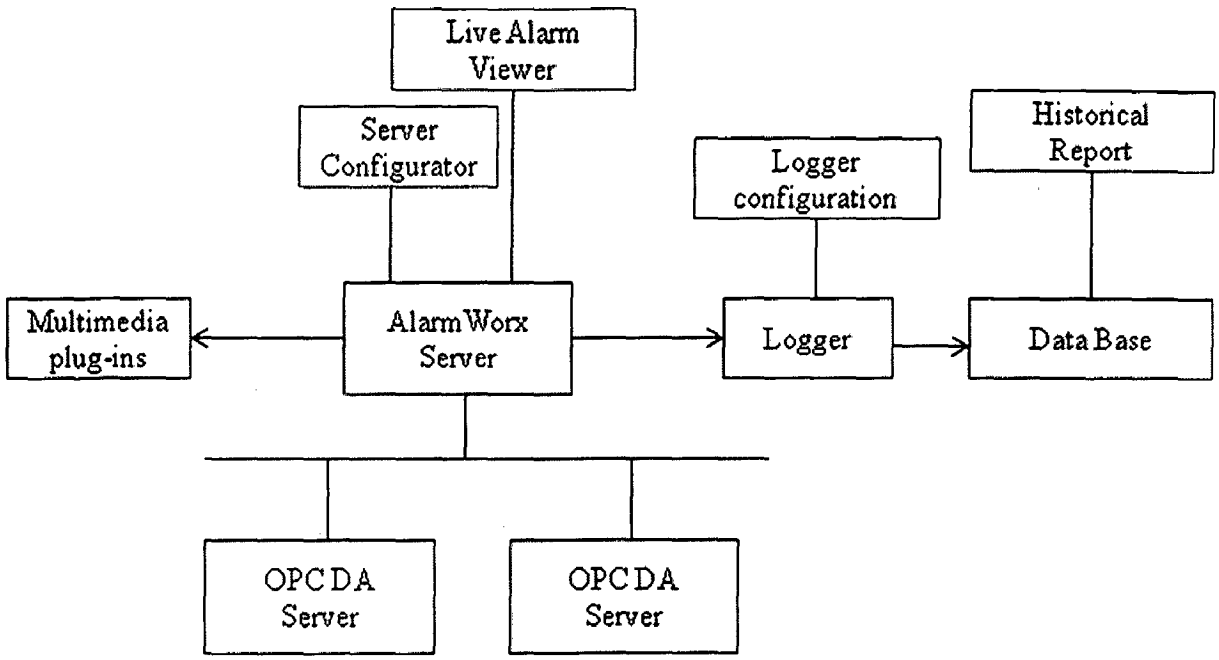


Fig 4.5 Architecture of AlarmServer

(e) Web-Server [4][28][38]

Smar WebHMI is a thin client Web solution that enables standard Web browsers, such as Microsoft Internet Explorer, for use as real-time operator interfaces to manufacturing and factory floor applications. Based on ActiveX technology, WebHMI provides you with a powerful and versatile approach to using the same standard HMI (Human Machine Interface) components included in ProcessView. WebHMI delivers industry-standard, real-time OPC (OLE for Process Control) information. WebHMI likewise delivers fast, worldwide operator graphical visualization, trending and alarming information—both real-time and historical—and HTML-based reports. Since WebHMI Web components are packaged in standard Microsoft .cab files server and clients can be located anywhere. Similarly, one can store .cab files anywhere on the network. Installed and resident on one or more WebHMI servers, these components (e.g., GraphWorX, TrendWorX, or AlarmWorX) are delivered automatically, quickly, and "in the background" to a browser on the client-side machine. Since WebHMI delivers the necessary components required for performing HMI and SCADA functions it is not necessary to have any servers and processview products installed

on the client machines. Ultimately, WebHMI turns a Web browser into an OPC client when the browser views Web pages located on any WebHMI server.

The ProcessView Web Publishing Wizard is used to publish the HMI files as web pages to the web server. It enables you to "export" your GraphWorX (.gdf), TrendWorX (.t32), and AlarmWorX (.a32) displays to HTML files and/or publish the HTML files to a Web server (LAN or Internet). In publishing displays to a Web server, WebHMI uses HTML to reference the files in an Internet-enabled format. Once a display is "exported" to an HTML file and then published to a Web server, client machines can browse it through an Internet browser, such as Microsoft Internet Explorer. Each display can be viewed as a Web page.

Interaction between clients and the WebHMI server is made possible by Smar GenBroker which uses TCP/IP communication over the Internet. The GenBroker Configurator allows to customize client/server architecture based on your network configuration.

4.2.4 OPC Clients

(a) Process View [27]

ProcessView suite is a set of powerful software modules in SYSTEM302 that includes all the best-of-breed applications the operator needs for process visualization and operation, advanced alarming, trend analysis, reporting, supervisory control, and much more. ProcessView is the base of the plant "information architecture" which provides the traditional monitoring functions. The operator can build the system and integrate workstations and other applications with unparalleled ease, economy and performance. The suit consists of Graph worx, Trendworx, Alarmworx and WebHMI.

(i) GraphWorx [27]

GraphWorX is a human-machine interface (HMI) software package for process control. GraphWorX is a fully compliant OPC client featuring ActiveX and OLE Automation technologies. GraphWorX includes simple and powerful object-oriented graphics design tools in the same application that makes it easy for the system integrator to create user-friendly graphics for the operators during the engineering stage. A view of graphworx screen is as shown in fig 4.5

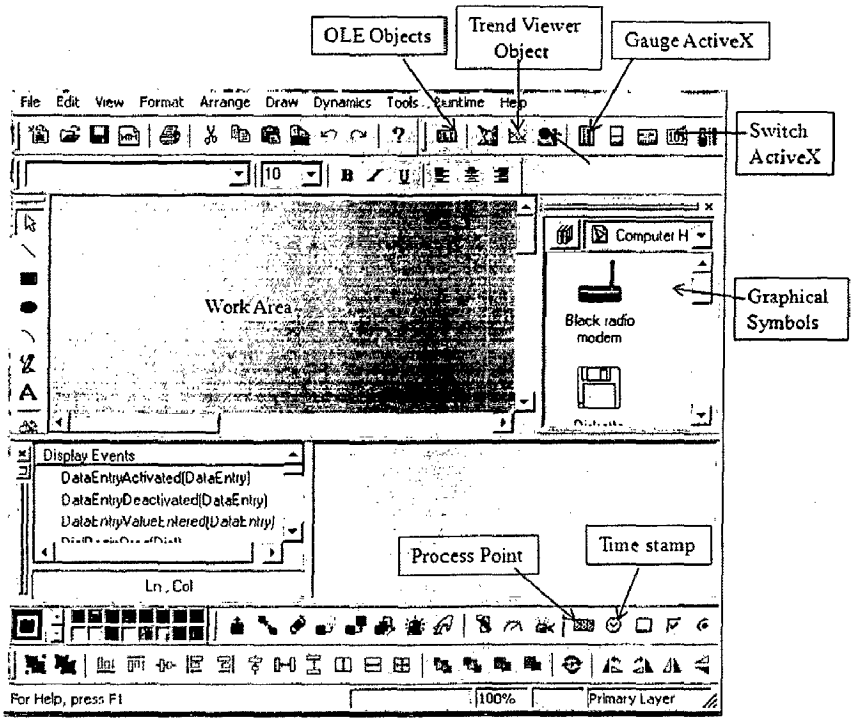


Fig.4.6 GraphWorX screen [27]

SYSTEM302 is completely tag-based, eliminating the need for mapping, cross-referencing and all worries about device and memory addresses associated with HMI in the past. The data in any server can be picked up simply by pointing and clicking in a universal tag browser, as shown in fig 4.6, without having to key in any tag or location. The browser shows all the registered servers in the local computer and on the remote computer. In order to access the servers on the remote server in the local network a set of registry settings called the DCOM settings and server registrations are required to be performed on all the remote clients.

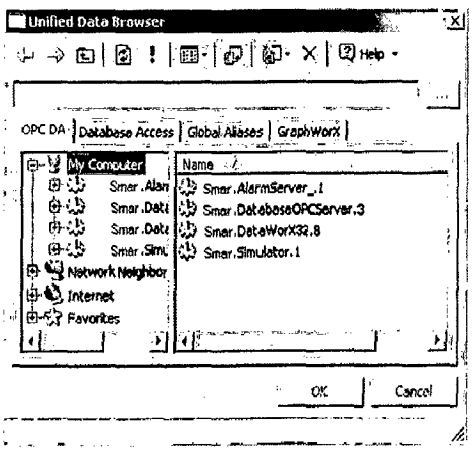


Fig 4.7 Unified Data Browser [27]

(b) LabVIEW OPC Client [39]

LabVIEW is graphical development platform for design, control and test. LabVIEW makes it easy to create human-machine interface (HMI) applications for remote monitoring and control. An OPC client program can be developed in LabVIEW software using the data socket. LabVIEW provides several activeX objects for developing a professional user interface such as graphs, knobs, switches etc. LabVIEW can be used as OPC client by connecting to an OPC server through a Data socket connection. Data socket has an OPC layer through which it can access the OPC custom interfaces of an OPC data access server. We can read and write to an OPC server using Data socket.

(c) OPC Automation Client in VB [8][37][36][39]

Visual Basic supports COM. There are two types of interfaces as discussed in section 3.2.5 namely, Custom interfaces and Automation interfaces. COM implementations from Visual Basic use what is called an "Automation" interface. The standard interfaces an OPC server supports are custom interfaces. Servers may or may not implement the automation interfaces. The automation clients cannot directly connect to the custom server. The "Automation Wrapper" DLL, provided by OPC Foundation should be used by the Automation clients to connect VB to any OPC data access server as shown in Fig.4.7.

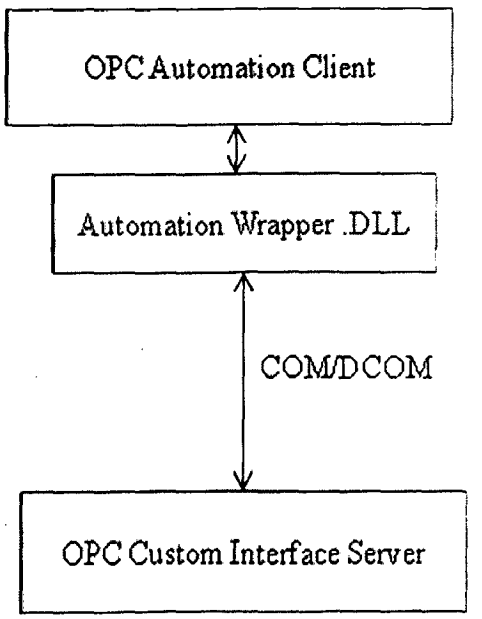


Fig.4.8. Custom and Automation Client Applications Interfacing to OPC Servers [8]

At higher level, an OPC Data Access Server is comprised of several objects: the Server, the Group, the Item and the Browser as discussed in section3.2.8. The OPC server object maintains information about the server and serves as a container for OPC group objects. The OPC group object maintains information about itself and provides the mechanism for containing and logically organizing OPC items. The Automation Wrapper connects to the OPC server and creates the groups and items in the server and gives you references to them in the VB program in an Object model that mirrors that of the server. A client registers its groups and items at the server. The server keeps the structure of all its clients. The "fully qualified item" is not sufficient to identify an item; a client may subscribe the same item in different groups. The pair ClientHandle, ServerHandle uniquely identifies an item. A handle is an identifier assigned in software to point to a memory location or data point. The sequences of steps involved in the development of an OPC client program in visual basic are given in the flow chart in Fig.4.9.

(d) Alarm Viewer [30]

The Alarm Viewer is a current-events alarm ActiveX. Because this component is an ActiveX, it can be placed in any ActiveX container application, such as GraphWorX, Microsoft Visual Basic, or a Web page. The Alarm Viewer displays current alarm information and handles the user interface to the alarm system (such as alarm acknowledgement). The layout of information displayed, including sort order, color, font, and displayed data, is user-configurable. One can be drop this ActiveX Control in the provided AlarmWorX Container, within any GraphWorX HMI Display, an HTML Internet/intranet-based Web page, or any other ActiveX container, and it automatically configures itself to deliver live alarms in a scrollable window. You can easily customize the view via its properties page to control the colors, fonts, columns, rows, alarm filtering, subscriptions, hot-links, etc.

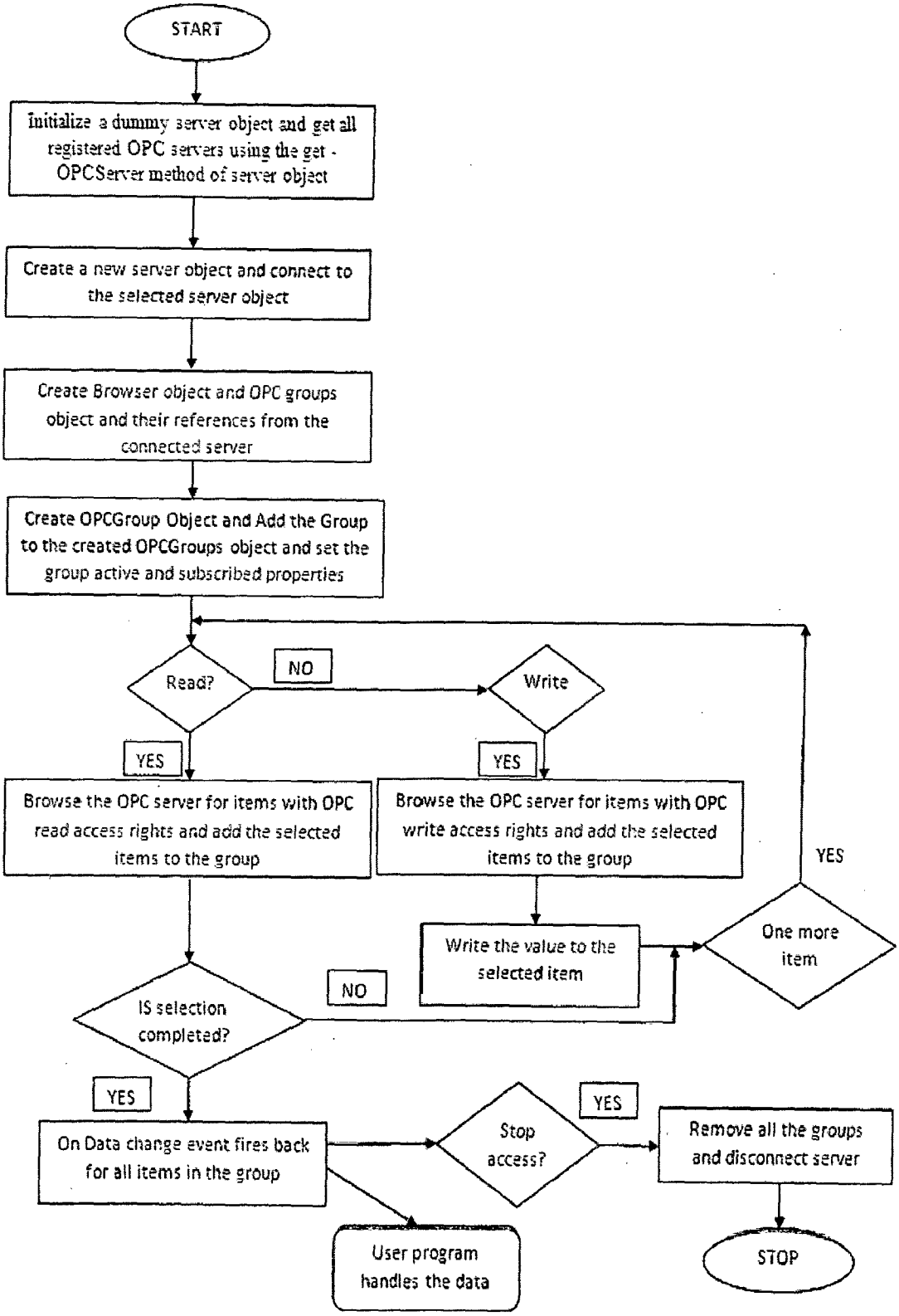


Fig.4.9 Flow Chart for VB Automation client

CHAPTER 5 Implementation of OPC Based Mixed-Mode SCADA Network

This chapter discusses of the implementation of developing a Mixed-protocol network with a Foundation Fieldbus protocol, one Modbus protocol and a proprietary DCON protocol and this shown as an application to a Small Hydro power station.

5.1 Model Hydro-Power Station

A Model Hydro power station consisting of two units is selected for designing a Mixed SCADA network. The layout of the model Hydro power station is as depicted in fig.5.1

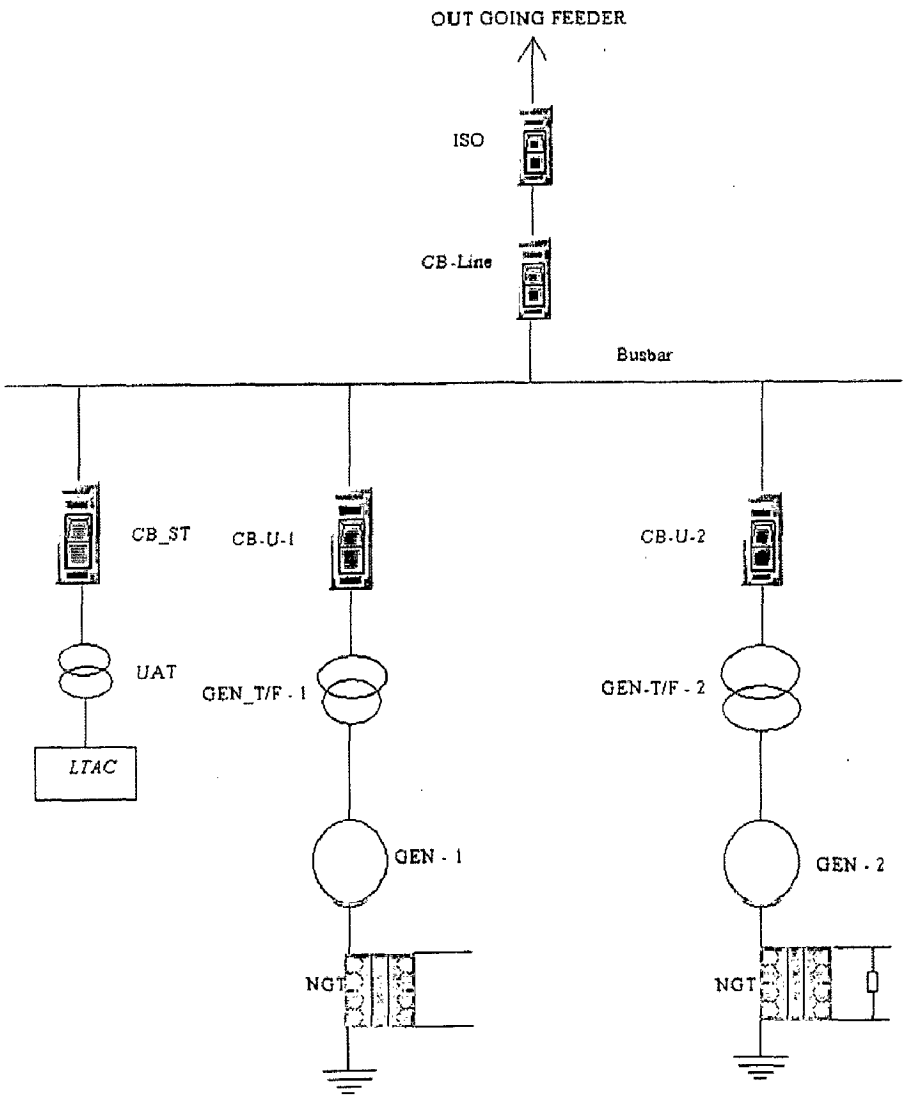


Fig.5.1 Single Line Diagram of Model Hydro-Power station

5.2 Network Architecture

The design of the implemented mixed-mode SCADA network implemented in the laboratory is depicted in the fig.5.2

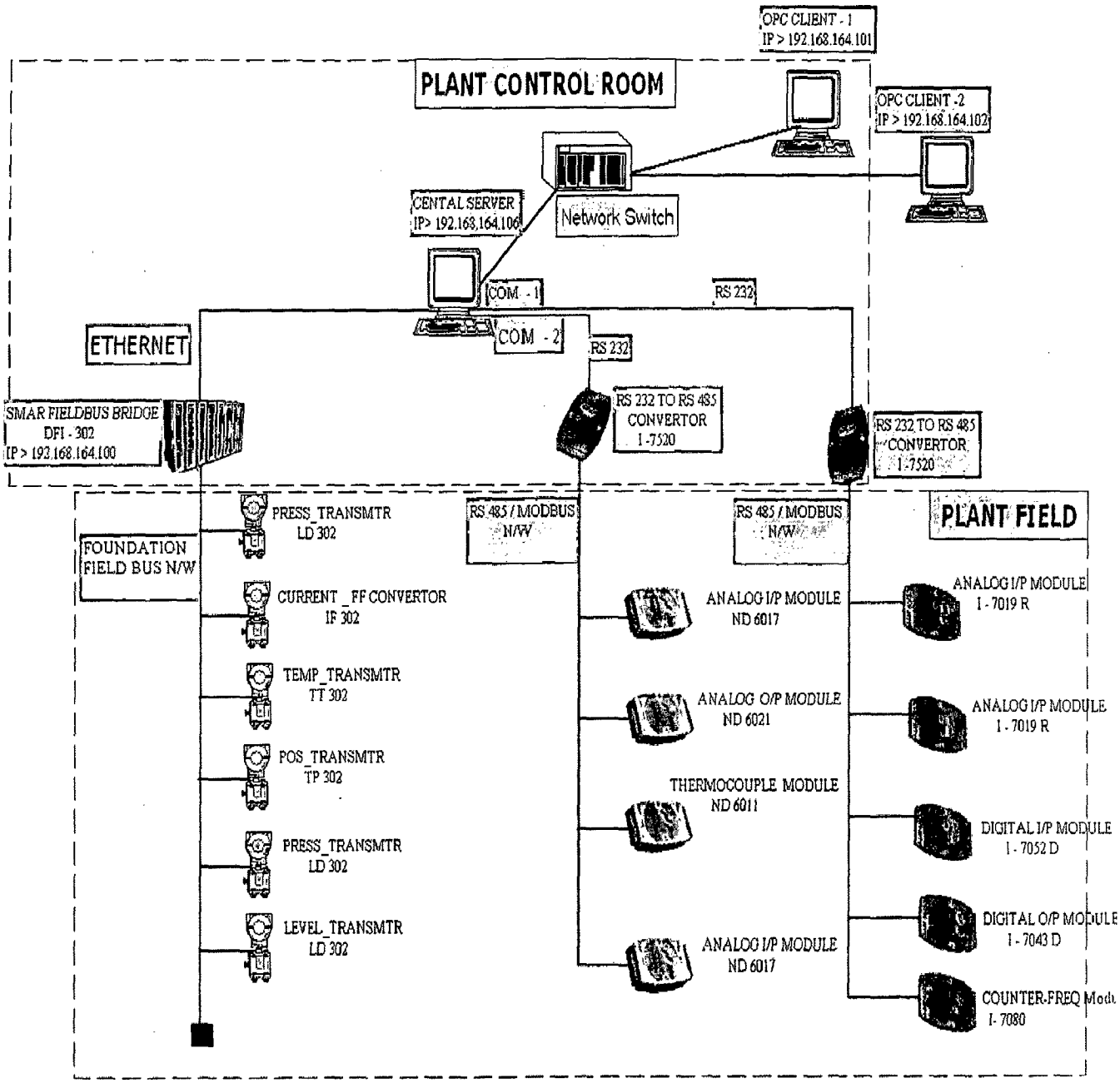


Fig. 5.2 Mixed-mode Network Architecture

The network has a central node (IP: 192.168.164.106) acts as a server station. This node connects the three networks, Foundation field bus network via a bridge (IP:192.168.164.100) and the two RS 485 networks through convertor modules. Syscon, the configuration software of Foundation field bus and Utility software for the configuration of

RS 485 modules are installed on this central node. The three OPC servers, smar DFI OLE server for Foundation fieldbus network, NAPOPC for I-7000 series modules and NDS-OPC server for ND-6000 series modules are installed on this central node. Smar WebHMI and hence Web-server are installed on this node and the HMI front panels are published as web pages and any remote client connected to this network on LAN can access these web pages through Microsoft Internet explorer. For demonstration of the function of remote monitoring through OPC two clients are connected in LAN as clients. The OPC clients, Graphworx, Trendworx, Alarmworx viewer are installed on the client nodes. For the clients to access the remote servers first of all the servers are to be registered on the client nodes, the access permissions and windows registries need to be configured on the client nodes as well as the server nodes. The detailed configurations steps and windows registry settings and the configuration steps for Foudation fieldbus and Modbus networks are explained in the below sections

5.3 Configuration steps

5.3.1 Foundation Field Bus Configuration

(A) Hardware Connection details

1. The bridge is powered with the help of single phase AC supply provided to its DF 50. The Power Supply for Backplane (DF50) is a high performance standard with universal AC input, 5 V DC (Backplane Power Supply) and 24 V DC (external use) outputs.
2. DF 52 provides DC power supply to the Fieldbus with a universal AC Input and 24 V DC isolated-output.
3. The Fieldbus Power Supply Impedance DF53 (4 ports) provides impedance for the power supply and the Fieldbus network, ensuring no short-circuit between the power supply and the communication signal on the Fieldbus.
4. DF 51 of the bridge or linking device is a powerful CPU module and also has an Ethernet port. An Ethernet cross cable is used to connect the bridge to the computer through its LAN port.

5. The transmitters LD 302, IF 302, TT 302, TP 302 are connected to the bridge through a daisy chain connection starting from the H1 Segment of the DF 51. In this way all the devices are connected in parallel to the power supply.
6. A terminator BT 302 is used at the end to avoid reflection of signal.

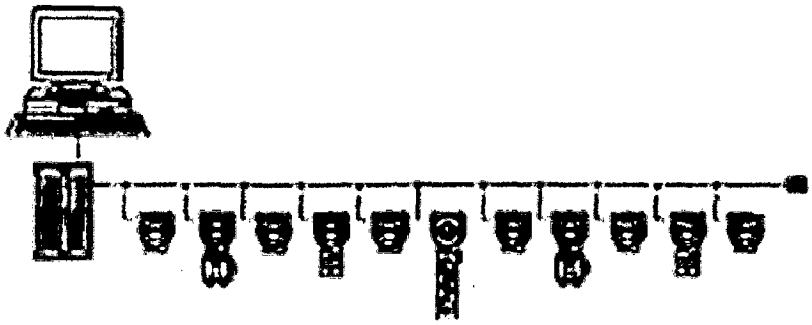


Figure 5.3 Connection Diagram [8]

In this way the devices are connected to form a Local Area Network.

(B) Setting up DFI 302

1. System 302 software is installed in the computer.
2. Hard keys (Hard lock protection) is used to get a DFI OLE Server License.
3. DFI302 working environment is composed of a network (Sub-Net) where IP addresses will be necessary for each connected equipment. The automatic solution for attribution of these addresses is called DHCP (Dynamic Host Configuration Protocol) Server. Using DHCP Server these IP addresses are generated automatically preventing any IP conflict between two distinct equipment. If the network does not have a DHCP server, DF51 will have the default IP address 192.168.164.100.Since the network to which the transmitters are connected did not have a DHCP server the IP addresses and Sub-Net Mask were to be set.
4. The IP address is modified to 192.168.164.XXX and the Sub-Net Mask is modified to 255.255.255.0. IP Address 192.168.164.100 is not used as this is already DFI302 default address.
5. To download the firmware we move to 'Dfi Download' dialog box by choosing 'DF 51' option from 'FB Tools Wizard'. However this firmware download is a onetime operation and need not be performed repeatedly.

(C) Creation of a New Project

The SYSCON software is run and a new project is selected. The DFI OLE Server is chosen from the communication menu. The H1 Segment is created and a new bridge is formed and the virtual field devices are attached to the bridge. Device tags are given to all the devices. The tags are exported to 'Taginfo' file in OLEServers folder using 'Export tags' option and the Communication is initiated using 'Init Communication'. The cross mark is removed by placing the device id in the attributes. In this way all the devices appear in the 'Live List'.

The SYSCON divides the actual application into two parts namely

1. The Logical Configuration.
2. The Physical Configuration.

(D) The Logical Configuration

The main Project window contains an 'Area' field which is a set of "Process Cells". Each process cell defines the application with Function blocks. Here the applications are named as 'control modules'. The actual control objective is built in the form of a control strategy by interconnecting the function blocks in the strategy window.

Since our application is only to monitor the field values in the slurry transportation plant no separate control strategy is required. Analog input blocks are used to display measured values in the form of percentages and arithmetic blocks are used to perform scaling operations.

(E) The Physical Configuration

H1 Segment Window deals with the Physical Configuration part of the application. It shows how the field devices are connected to the actual H1 Fieldbus segment and finally to the Linking device or the bridge. The physical configuration starts with creation of a H1 Fieldbus segment connected to a linking bridge and wiring all the devices to that Fieldbus segment. In this manner all the transmitters are instantiated using SYSCON.

Double clicking on the respective device displays the list of function blocks a field device. For any device four blocks are compulsory. They are namely

1. Resource block
2. Transducer block.

3. Display block.
4. Diagnostics block

Resource block

Resource blocks are used to define hardware specific characteristics of function block applications. Similar to transducer blocks, they insulate function blocks from the physical hardware by containing a set of implementation independent hardware parameters.

Transducer block

Transducer blocks insulate function blocks from the specifics of I/O devices, such as sensors, actuators, and switches. Transducer blocks control access to I/O devices through a device independent interface defined for use by function blocks. Transducer blocks also perform functions, such as calibration and linearization, on I/O data to convert it to a device independent representation. Their interface to function blocks is defined as one or more implementation independent I/O channels.

Diagnostics block

This transducer block provides the following features:

- Online measurement of block execution time
- Hardware revision
- Firmware revision
- Serial number of device
- Serial number of main board.

Display Transducer Block

The display transducer is responsible to show on the LCD screen, one chose variable when it is in monitoring mode or a configured menu when in local adjustment mode. The display transducer is completely configured via SYSCON. It means the user can select the best options to fit his application. Among the possibilities, the following options can be emphasized: Mode block, Outputs monitoring, Tag visualization and Tuning Parameters setting. The user, when configuring, may select up to seven parameters of any block, executing in the local device. It means that the device itself is executing that Display Transducer Block.

The 'Target' parameter of the 'MODE BLK' of all the blocks is set to 'Auto'.

The Tag Designations for all the blocks are given.

(F) Configuring Transducer block

Calibration:

Two types of calibration can be performed on the transmitters

1. Calibration with reference: This is used to adjust a transmitter's working range using a standard as reference.
2. Calibration without reference: This is used to adjust the transmitter working range where the user specifies the limit values.

Type of calibration done depends on the transmitter and its area of application.

The Upper Trim and the Lower Trim is configured via SYSCON using CAL_POINT_LO and CAL_POINT_HI. A convenient engineering unit should be chosen before starting the calibration. This engineering unit is configured by the CAL_UNIT parameter.

For calibrating the device, the lowest input has been applied to the transmitter and the corresponding reading obtained in the transmitter is set in CAL_POINT_LO. Similarly the highest input has been applied to the transmitter and the corresponding value is placed in CAL_POINT_HI. In the similar manner calibration can also be done at the intermediate values.

Calibration can also be done using Local Adjustment. The lower and upper values set must be within the sensor range specified.

There are many other parameters like Characterization Trim, Temperature Trim etc. which can be modified depending on the application. The mnemonics RW against a parameter indicates that the corresponding parameter can be modified depending on the application.

(G) Configuring Analog Input Block

An analog input bock is created for every device. The Analog Input block takes the input data from the Transducer block, selected by channel number, and makes it available to other function blocks at its output.

Transducer scaling (XD_SCALE) is applied to the value from the channel to produce the FIELD_VAL in percent. The XD_SCALE engineering units code and range must be suitable to the sensor of transducer block connected to the AI block.

The L_TYPE parameter determines how the values passed by the transducer block will be used into the block. The options are:

Direct - the transducer value is passed directly to the PV. Therefore OUT_SCALE is useless.

Indirect - the PV value is the FIELD_VAL value converted to the OUT_SCALE.

Indirect with Square Root - the PV value is square root of the FIELD_VAL converted to the OUT_SCALE. Direct option is selected in all the analog input blocks. Square root option can be selected when we wish to see the value of the flow directly on the device. Then the proper scaling of the parameter can be done using arithmetic block. The ARTH block is intended for use in calculating measurements from combinations of signals from sensors. Similarly other blocks like Analog Output block, PID block etc. can be used to frame a strategy depending on which the entire process can be controlled.

(H) Configuring Display Block

This block supported by devices with LCD display can be used to monitor and actuate in local parameters of blocks. This block can configure seven different displays which can be seen on the LCD display.

Parameters Configured:

1. BLOCK_TAG_PARAM: This is a tag of the block to which the parameter belongs to use up to a maximum of 32 characters. This parmeter is filled in with the designations of the transducer and analog input blocks from where we get the required parameters to be monitored.
2. INDEX_RELATIVE: This is the index related to the parameter to be actuated or viewed. The primary value or the measured variable has an index relative of 14 in the transducer block and an index relative of 7 in the analog input block.
3. SUB_INDEX: This is the offset of the parameter which we wish to monitor. The primary value or the measured variable has a sub index relative of 2 in both the transducer block and analog input block.

4. MNEMONIC: This is mnemonic of the value which can be displayed on the screen. It is generally set as P_VAL or the name of the unit of the primary value.
5. INC_DEC: It is the increment and decrement in decimal units when the parameter is Float or Float Status time, or integer, when the parameter is in whole units. It is set to 0.01 or 0.25.
6. DECIMAL_POINT_NUMBER: This is the number of digits after the decimal point (0 to 3 decimal digits). The required precision can be set on the display. This is set to 2.
7. ACCESS: The access allows the user to read, in the case of the "Monitoring" option, and to write when "action" option is selected, and then the display will show the increment and decrement arrows. Monitoring option is selected.
8. ALPHA_NUM:

   These parameters include two options:

   1. value
   2. mnemonic.

   In option value it is possible to display data both in the alphanumeric and in the numeric fields, this way, in the case of a data higher than 10000, it will be shown in the alphanumeric field. Mnemonic option is selected.

All these parameters are configured for each display that is presented on the transmitters.

After configuring all the devices offline, Tags are again assigned and exported and download operation is performed to set up the transmitters with the required configuration. 'Partial download' can also be performed to the individual devices. Now the devices acquire 'Good' online characterization.

The real time change in the measured variable can be seen both in transducer block 'primary value'. The devices can also be reconfigured online by modifying the values online. They can also be offline characterized in which case the Tags have to be exported and the configuration needs to be downloaded again to acquire 'Good' online characterization. The following figure.5.4 shows the configured SYSCON window.

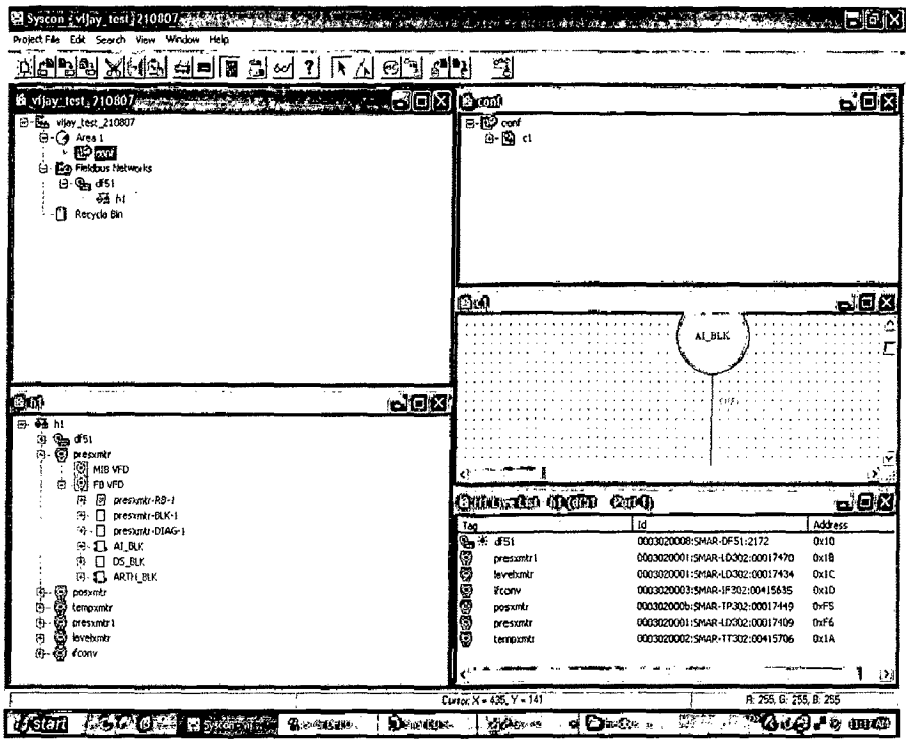


Fig.5.4 Syscon configuration window

5.3.2 Configuration of ICP DAS Modules

The I-7000 utility and NAPOPC server are used to configure the I-7000 series modules. The modules are connected to the RS 485 network in a multi-drop fashion and the network is connected to the comport-2 of the host computer as shown in fig.5.5.

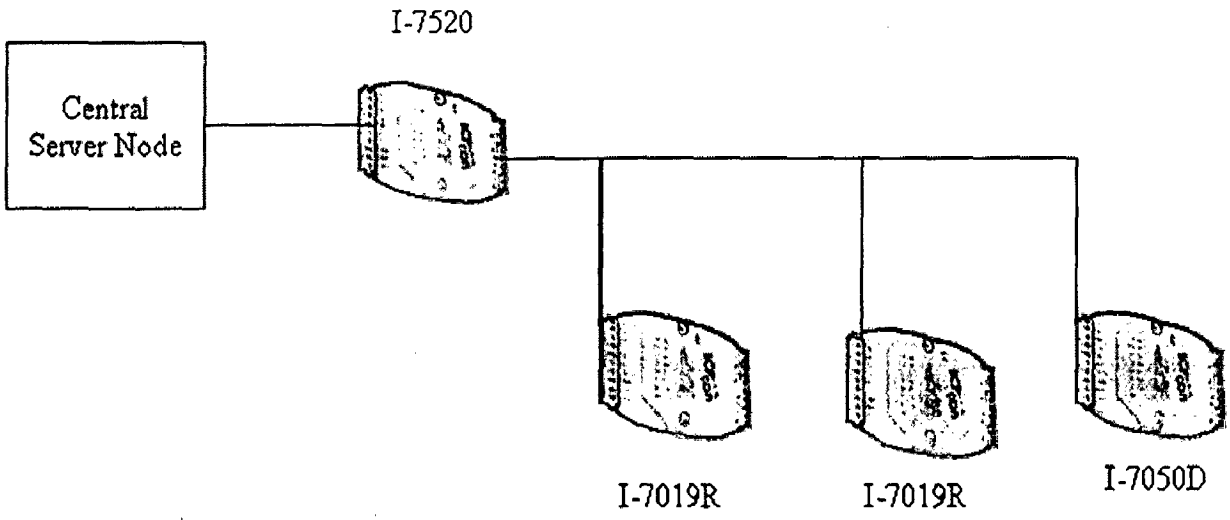


Fig.5.5 Multi-drop RS485/ MODBUS Network (ICP DAS)

The following are the steps to configure the RS 485 modules:

1. Start the NAPOPC server from start->All programs-> NAPOPC server.exe.
2. Then "search modules" function as shown in fig5.5 configures the OPC server. It searches the RS485 network for connected modules and the modules are added.
3. The monitor function is used to view the real-time values of the tags.

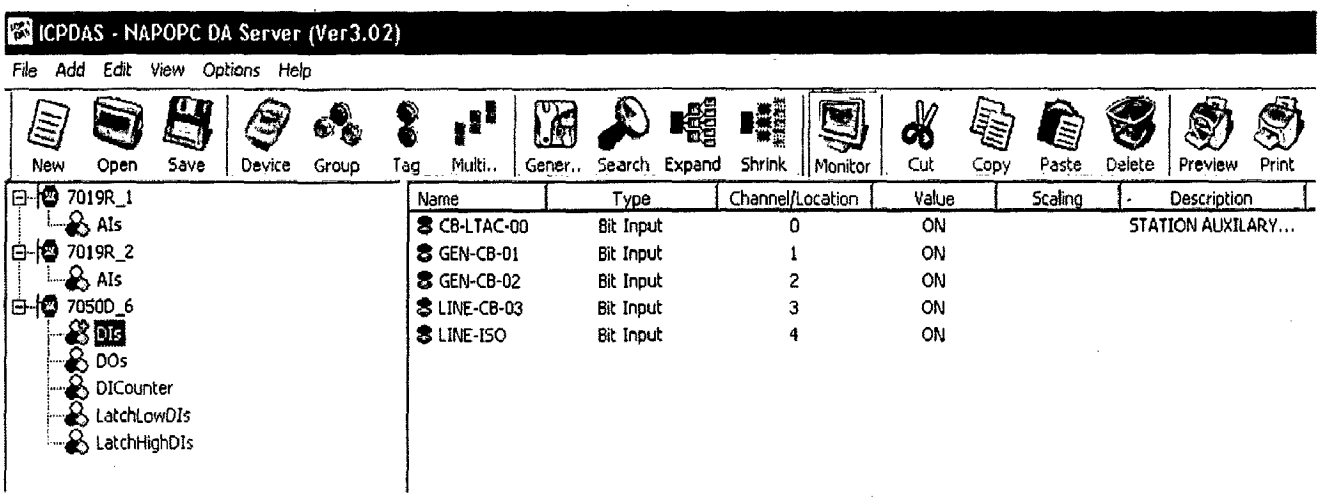


Fig.5.6. Fully Configured NAPOPC DA server

5.3.3 Configuration of NuDAM modules

The NDSOPC server is used to configure the ND-6000 series modules. The modules are connected to the RS 485 network in a multi-drop fashion and the network is connected to the comport-2 of the host computer as shown in fig.5.7.

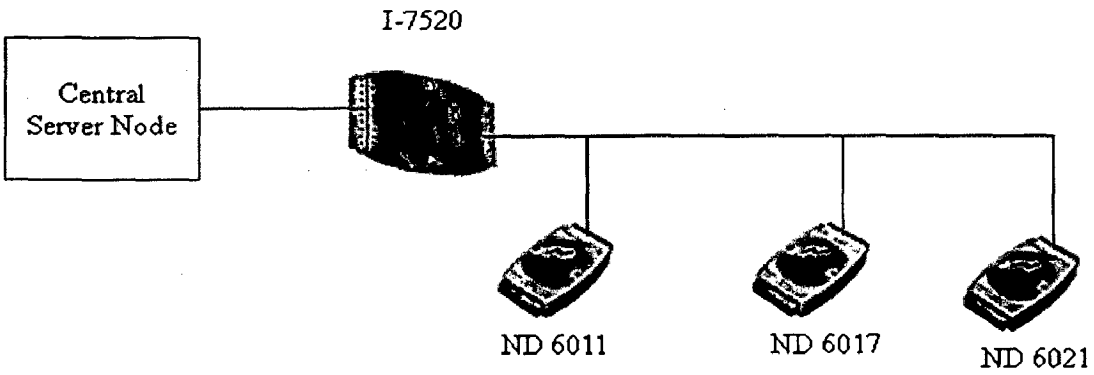


Fig.5.7. NuDAM RS485/MODBUS Network

The following are the steps to configure the RS 485 modules:

1. Start the NDSOPC server from start->All programs-> NAPOPC server.exe.

2. Then "search modules" function as shown in fig5.8 configures the OPC server. It searches the RS485 network for connected modules and the modules are added.

3. The start diagnostic function is used to view the real-time values of the tags

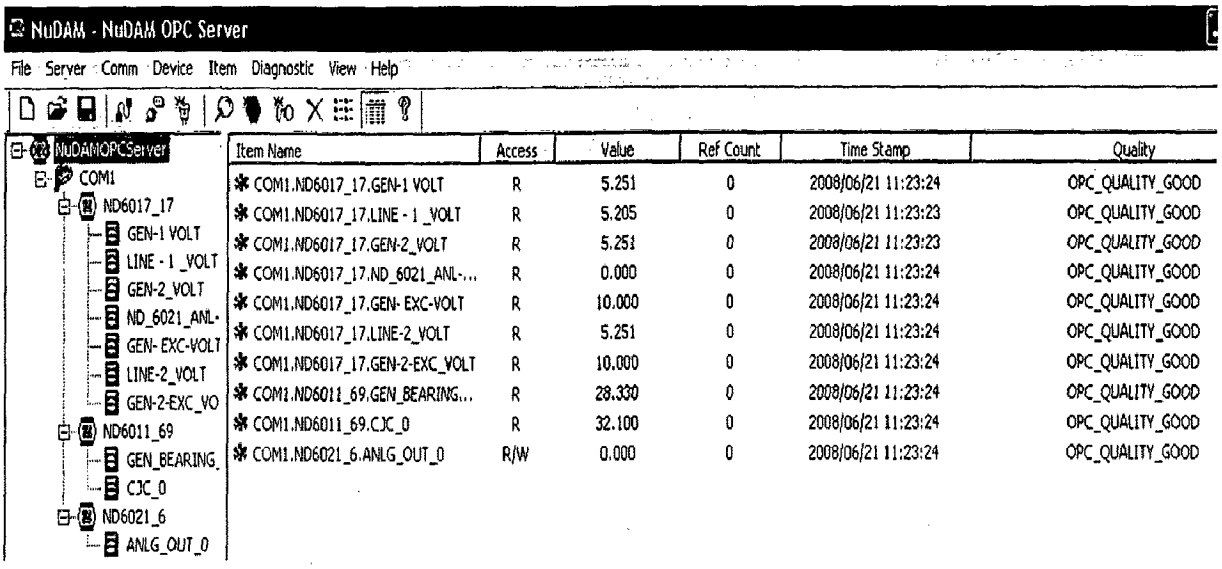


Fig.5.8. Fully Configured NDS OPC server

5.4 SCADA Functions Implemented

The following SCADA functions are implemented as a part of physical simulation of a mixed-mode SCADA network for a model Hydro-power station.

5.4.1 Data Acquisition

The following real-time values of the following parameters are acquired through these networks

Intake Pressure Head – Pressure Transmitter (LD 302)

Level measurement

at fore bay - Level Transmitter (LD 302)

Level measurement

at tailrace - Level Transmitter (LD 302)

Speed /Frequency - Counter Frequency module I -7080D

OPU Pressure - Pressure Transmitter (LD 302)

Inlet Nozzle Position - Position transmitter (TP 302)

Generator Parameters	Generator Temperatures
Voltage Phase-A	Stator core Temperature Ph – A - 1
Voltage Phase- B	Stator Core Temperature Ph – A - 2
Voltage Phase –C	Stator Core Temperature Ph – B - 1
Current Phase –A	Stator Core Temperature Ph – B - 2
Current Phase –B	Stator Core Temperature Ph – C - 1
Current Phase – C	Stator Core Temperature Ph – C - 2
DC Excitation Voltage	Bearing Temperature BT - 1
DC Excitation Current	Bearing Temperature BT – 2

Generator Transformer Parameters	Generator Transformer temperatures
Current Phase – A	Transformer Oil Temperature
Current Phase – B	Winding Temperatures
Current Phase - C	

Unit Auxiliary Transformer Parameters	Digital In and Out
Current Phase – A	Unit -1- CB
Current Phase – B	Unit – 2 – CB
Current Phase – C	Feeder CB
Transformer Oil Temperature	Station Isolator
Transformer Winding Temperature	

The developed Mixed-mode network is simulated for a Model Hydro power station. Hence the above parameters of the power station are simulated with the DC voltage sources and current sources, RTDs, Thermocouples and other available sensors, so that the test of the developed Mixed-mode SCADA network can be successfully carried out.

5.4.2 Remote Monitoring

The central server station is connected in LAN with two client nodes as shown in fig.5.2 OPC client applications are installed on the client nodes. The clients and server are required to be configured with the following DCOM settings.

DCOM Settings

Two different configurations are to be done, the client-side one and the server-side one. In the client-side you may have an end-user program like Syscon and some components of Smar OLE Server software (CONFPrx.dll, IProxy.dll and OPCProxy.dll files, and the required information to NT registry). In the server-side you must have the whole Smar OLE Server software in order to establish communication between software client(s) and Hardware plugged in the computer. The DCOM settings vary with the operating system under use. The below settings mentioned in this chapter are Windows XP service pack2 specific since all the nodes have the applications running in this operating system. The major goal of Windows XP Service Pack 2 is to reduce common available scenarios for malicious attack on Windows XP. The Service Pack will reduce the effect of most common attacks in four ways:

1. Improvement in shielding Windows XP from the network
   a. RPC and DCOM communication enhancements
   b. Enhancements to the internal Windows firewall
2. Enhanced memory protection
3. Safer handling of e-mail
4. Internet Explorer security enhancements.

Most OPC Clients and Servers use DCOM to communicate over a network and thus will be impacted due to the changes in Service Pack 2. When Service Pack 2 is installed with its default configuration settings, OPC communication via DCOM will cease to work. Since the callback mechanism used by OPC essentially turns the OPC Client into a DCOM Server and the OPC Server into a DCOM Client, the instructions provided here must be followed on all nodes that contain either OPC Servers or OPC Clients.

(A) Configuring the Firewall

The Windows Firewall allows traffic across the network interface when initiated locally, but by default stops any incoming "unsolicited" traffic. However, this firewall is "exception" based, meaning that the administrator can specify applications and ports that are exceptions to the rule and can respond to unsolicited requests.

By default the windows firewall is set to "On". This setting is recommended by Microsoft and by OPC to give your machine the highest possible protection. For trouble shooting, firstly the firewall is turned off.

(B) Configuring your Network Hosts

There are two possibilities when configuring your machines to be involved in DCOM communication. You can use only Workstations (standalone) or Workstations in a Domain Note that any NT Server may be or not a server-side machine for the PCI OLE service. The advantages of one over another may depend on your local network architecture. Both processes require help of your network administrator. To choose which one to use remember that Domain based architecture has a single security database and thus is the simplest way. In laboratory, a client-server network based on standalone workstations is implemented.

First a work group "OPCGROUP" is created on the local area network. On each machine in LAN a new user is created and added to the so created OPC GROUP.

(C) System-wide settings

1. In the local security settings (Control panel\Administrative tools\Local security policies\local security settings) navigate to the Network access: Sharing and security change the option to local users authenticate as themselves.

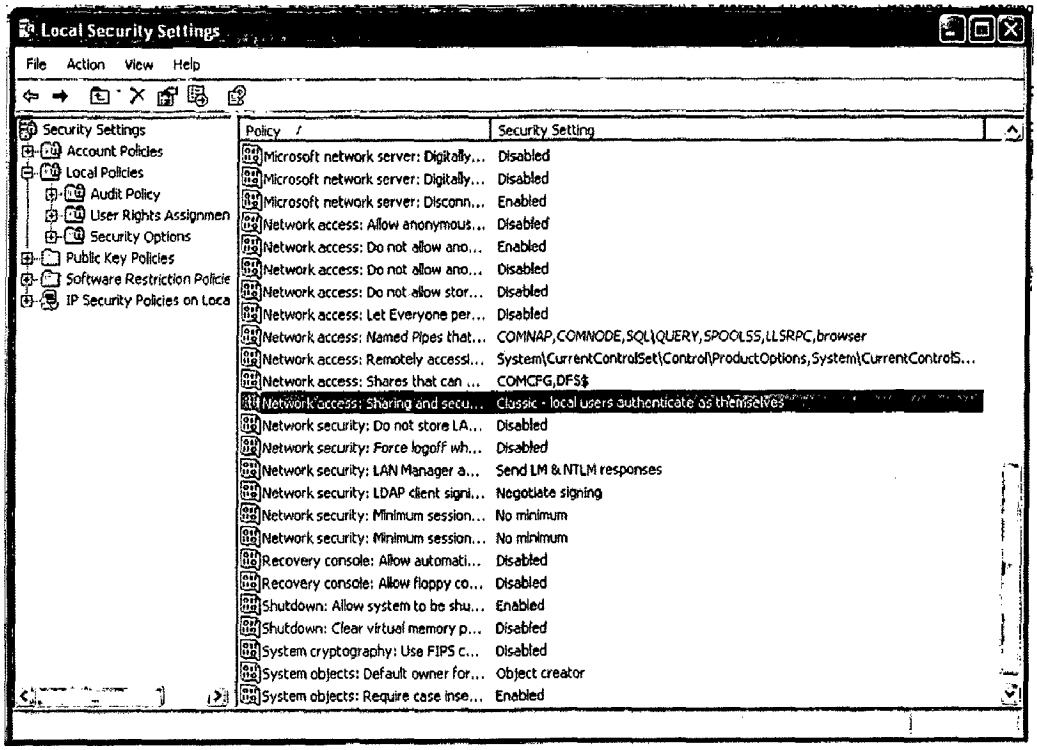


Fig 5.7 Local security settings

2. Go to Start -> Run and type DCOMCnfg and click on OK.

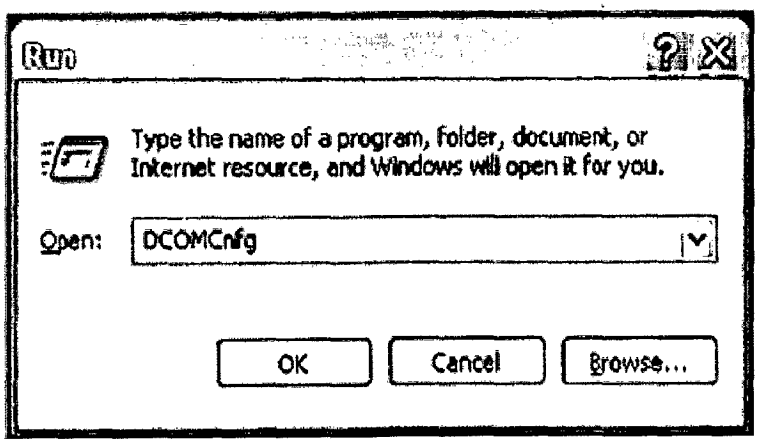


Fig 5.8 DCOMCNFG in run command

Click on **Component Services** under the Console Root to expand it.

3. Click on **Computers** under Component Services to expand it.
4. Right click on **My Computer** in the pane on the right and select Properties
5. Go to the COM Security tab as shown in fig.5.9 and note there are four permission configurations which have to be edited:

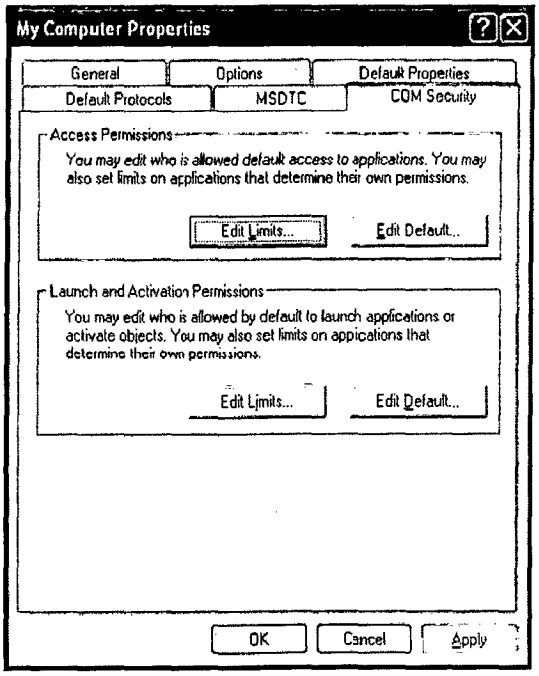


Fig 5.9 My Computer Properties page in DCOM configurations

6. Edit the Limits for Access and Launch

a. Access Permissions – **Edit Limits...**

You need to check the Remote Access box for the user labeled ANONYMOUS LOGIN in this dialog.

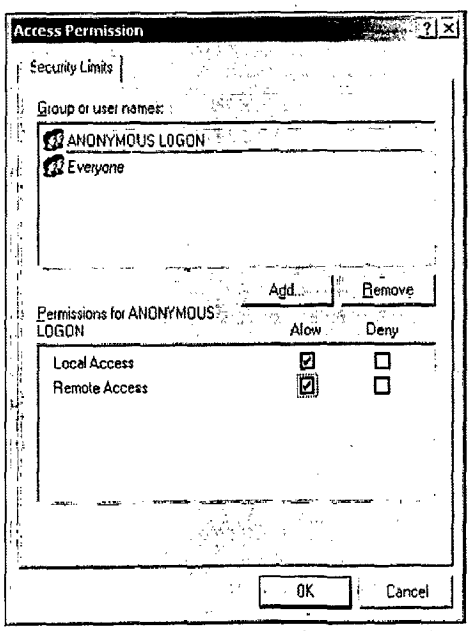


Fig.5.10 Access permissions Tab

This setting is necessary for **OPCEnum.exe** to function and for some OPC Servers and Clients that set their DCOM 'Authentication Level' to 'None' in order to allow anonymous

connections. The OPC Enum.exe is OPC server enumerator. It searches the registered OPC servers on the computer.

b. Launch and Activation Permissions – **Edit Limits...**
You need to check the remote boxes for the user labeled Everyone in the dialog as shown in fig.5.11

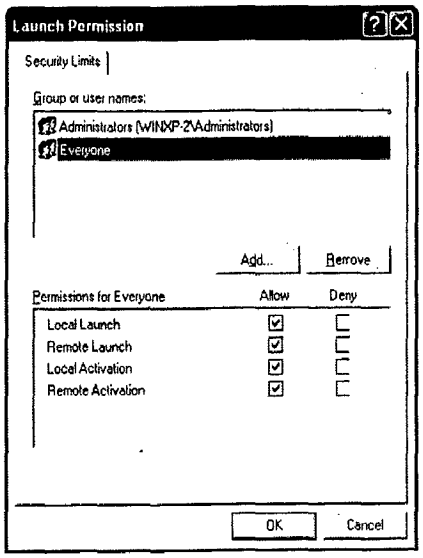


Fig.5.11 Launch permissions Tab

7. Edit Default Permissions for Access and Launch
For each user (or group) that participates in OPC communication (e.g. "OPC Users"), both the **Local Allow** and **Remote Allow** checkboxes are both checked.
8. Select the Default Properties folder and set the following fields:
8.1. _ Enable Distributed COM on this computer.
8.2. Default Authentication Level: **Connect**.
8.3. Default Impersonation Level: **Identify**.

(D) Server Specific settings

The system wide settings are performed on all the nodes in LAN and will be same for all the nodes.

1. From DCOMCNFG properties (Run: dcomcnfg->componentservices->My computer >dcomcnfg) Select the Applications folder and double click on specific server application.
2. Select the Location folder and check Run application on this computer option.
3. Select now the Security folder. Check the option Use default access permissions and Use default launch permissions.

(E) Client Specific Settings

The system wide settings will similar to that of the server system wide settings. A part from these settings following settings has to be made on individual client nodes.

1. Go to Start -> Run and type DCOMCnfg and click on OK.
2. Select the applications folder (->component services->computer->My computer->dcomcnfg)

double click on each server icon and select the location folder and give the node id of the central server where the server application runs as shown in fig.5.12.

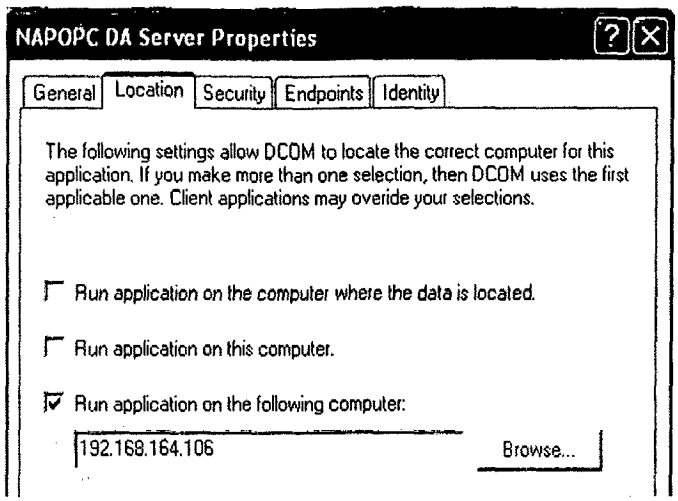


Fig.5.12 Server application properties Tab

3. Select now the Security folder. Check the option Use default access permissions and Use default launch permissions.
4. Repeat these steps 1 to 3 for all the OPC servers. Here these steps are repeated for the three servers Smar dfi OLE server, NAPOPC server, NDS OPC server.

After fully configured, the client nodes will be able to access all the OPC servers on the server station. The Graph Worx software is used to develop the professional Human Interface front panels for the workstation of a Small Hydro power station. Two other test clients developed with standard OPC interfaces in LABVIEW and Visual Basic 6.0 are tested for remote access.

5.4.3 Alarm and Event Notifications

The Alarm worx server is installed on the central server station. The AlarmWorX Server receives field data from OPC Data Access servers, Smar DFI OLE, NAP OPC and NDSOPC servers and performs alarm detection based on the conditions specified, may be on

a single tag or an expression comprising of multiple tags, in the configuration file and report to the alarm viewer installed on the client nodes . The AlarmWorX Viewer and the AlarmWorX Logger are used as two clients to receive these notifications from the server on the remote client nodes. The alarm conditions are set during the configuration of the server for individual tag.

5.4.4 Alarm and Event logging

The AlarmWorX Logger provides a permanent copy of alarm and event notifications produced by any OPC Alarm and Events server, including the AlarmWorX Server. The Logger Application (AWXLog32.exe) that provides the runtime storage and printing has no user interface and may optionally be run as a service. The logger typically reads its configuration information from a Microsoft Access. The Alarm Logger Configurator (AWXLogCfg.exe) is used to make changes to the database file that the Logger uses for configuration information.

5.4.5 Control Functions

The main control function is to remotely open or close the circuit breakers from a control station. In the circuit breakers two contacts namely main contacts and auxiliary contacts exists. Both the contacts are connected to the same lever. Both the contacts move simultaneously, main contacts are connected to the line and we monitor the auxiliary contacts. This function is simulated with a two change over relay. The relay has two contacts, main contacts and auxiliary contacts. The main contacts are connected to a light load and the auxiliary contacts are monitored through a digital input module. The digital out module is connected to the relay such that when a command '1' is written through the module the relay is excited and the contact is closed to supply load. Therefore from a remote client the position of the contacts is monitored and controlled. The important control function is to support the interlocking of the Line circuit breaker and isolator. The interlocking should be such that when we try to close the circuit breaker when the isolator is open the command should not be implemented in fact a message should be given to the operator and if we open the isolator when the circuit breaker is closed then it should not open the isolator and intimate

the operator. That is the circuit breaker must open the circuit or close the circuit not the isolator. This interlocking functionality is implemented.

5.5 WEB Monitoring

Smar WebHMI application is used to publish the HMI front panels as web pages. First the Internet Information service 6.0 is installed on the central server station. Then the WebHMI application is installed and the name of the web server and node ID are configured during the installation. The architecture of the web server and client communication implemented is as shown in figure.5.3.

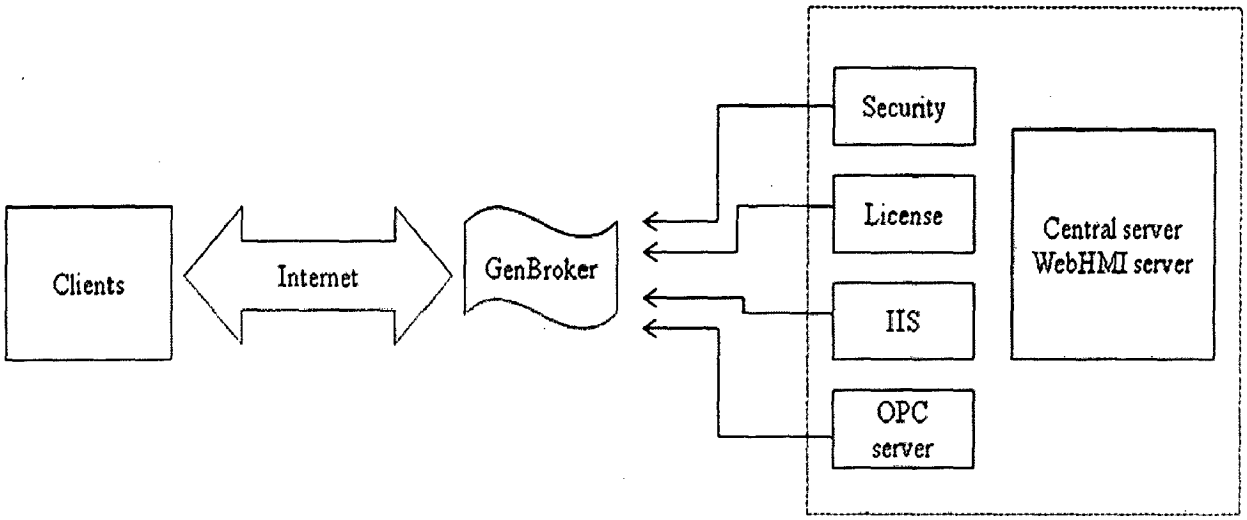


Fig.5.3 Architecture of WebHMI server and Client communication

The webHMI server communicates to thin clients through the genbroker provided as an integral part of process view. GenBroker uses TCP/IP and SOAP/XML channels to achieve real-time and secure communications between Web browser clients and WebHMI servers. Genbroker is configured to enable the communication over the internet. It acts as a bridge between Web HMI server and web clients. A thin client sends out a request over the Internet to the WebHMI server. The thin client uses GenBroker to transmit the request. The WebHMI server's response to the request is also returned via GenBroker. But if the thin client's request requires a response from a remote server in a local area network (LAN), WebHMI could be configured to use DCOM. The communication type in configuration of the genbroker is specified as DCOM over TCP/IP. Since the application uses the OLE and

activeX technologies it can transfer the configuration files to the web client during the runtime. Hence the client even does not require any process view components but as the communication channel used is DCOM over TCP/IP it requires DCOM configuration settings to be performed on the clients as well.

The configurations of the security server, software licensing, alarm server and genbroker are maintained as Microsoft .cab files which can be transferred to the remote client during the runtime. First of all the HMI front panels contain the visual basic components which can not be transferred over the internet hence the pages should be saved as non-VBA pages in the working directory of the WebHMI application (C:\ Inetpub\WWWroot\WebHMI\ will be by default). The web pages are now published using the web publishing wizard of Smar WebHMI application. Now typing the URL of the web page in the internet explorer. The syntax for URL is http://<WebServer name>/<Webpage name>.html

The mixed-mode network is successfully deployed and the two client nodes are successfully communicated with the server.

The following figure 6.1 shows the implementation of the network in the laboratory.

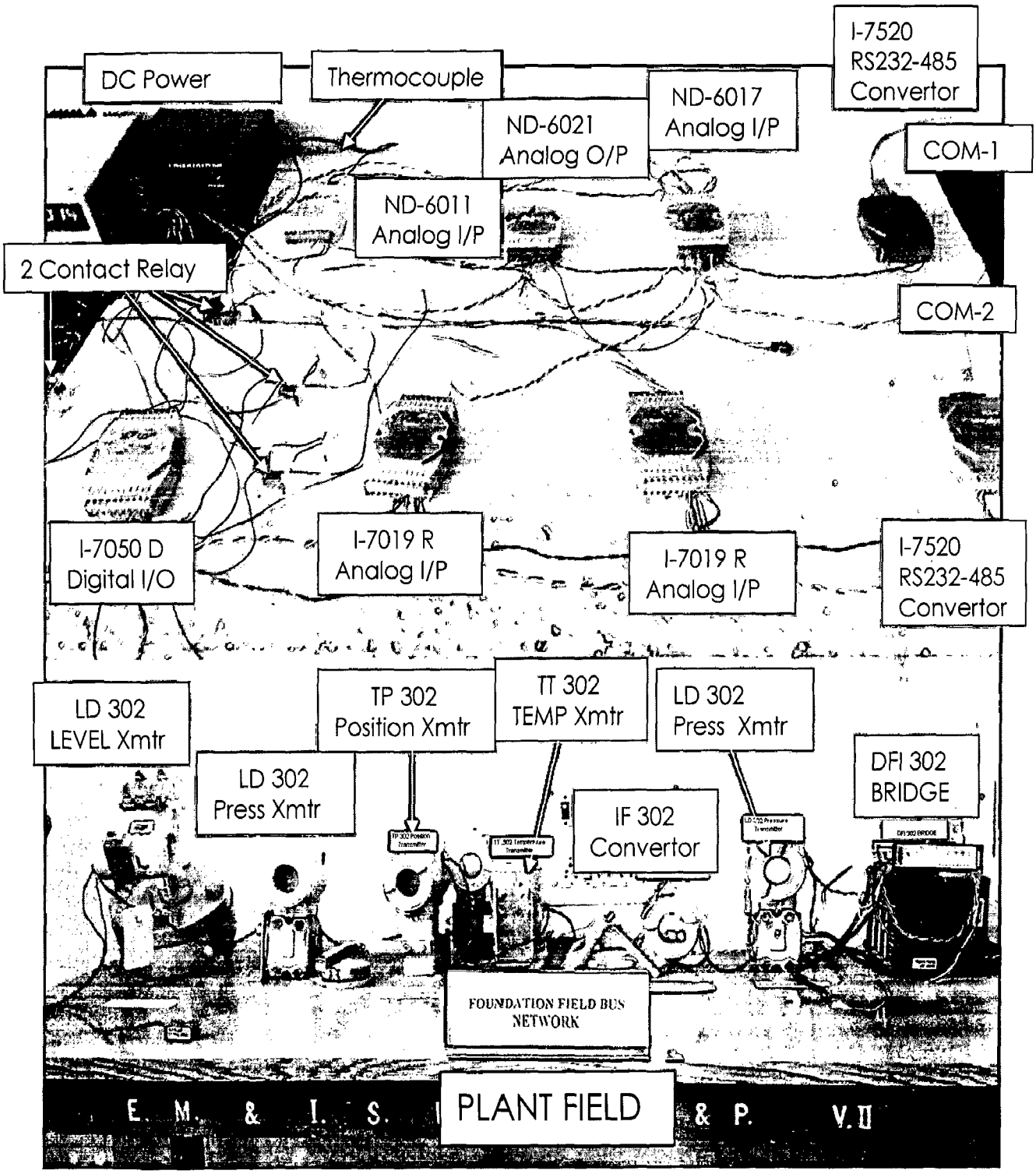


Fig 6.1 Foundation FieldBus and MODBUS Networks Implemented in Laboratory

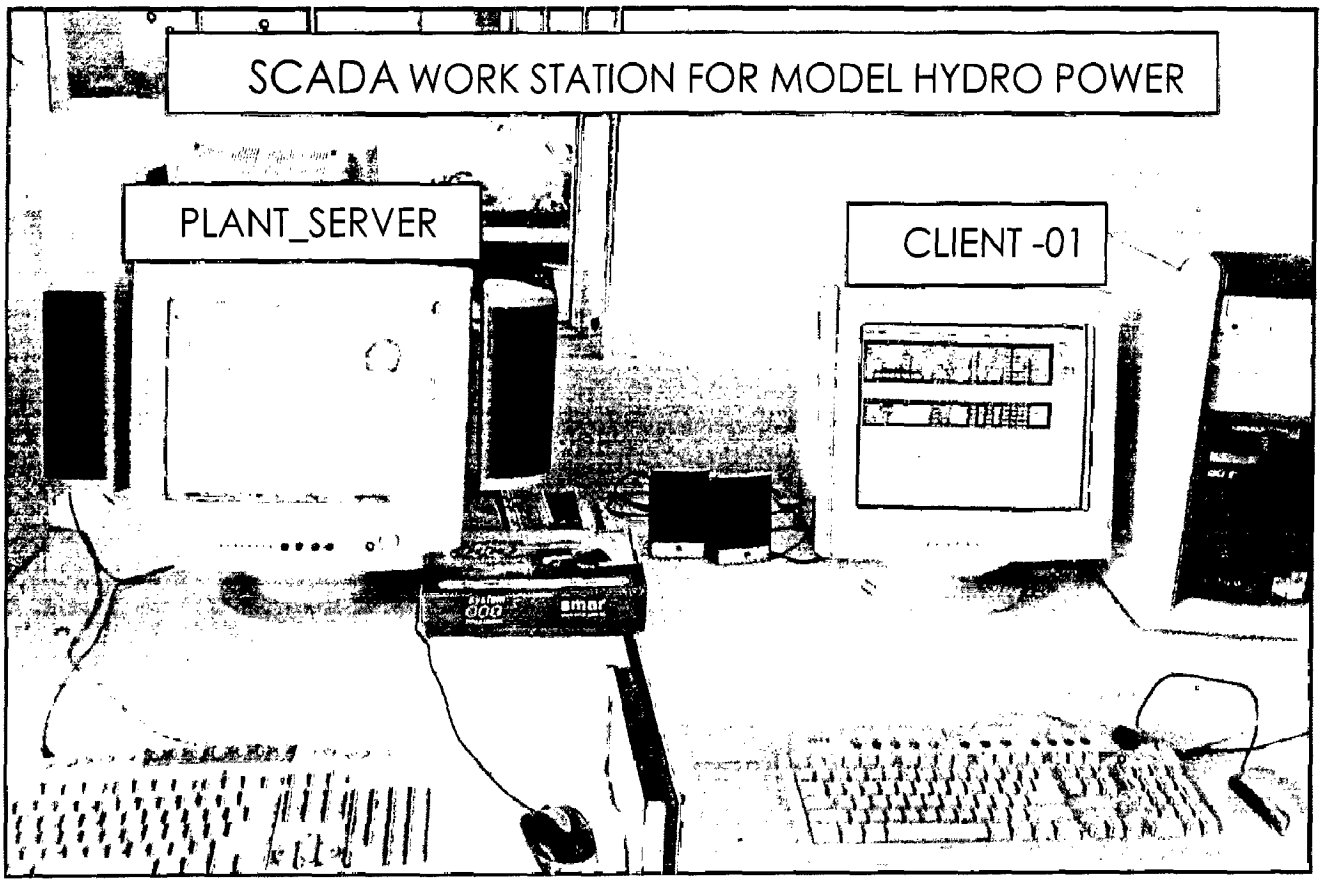


Fig 6.2 Server and Client Connected in LAN

The following figure shows the Unified data browser, used to browse the servers and tags in local node and in the network as well. Figure 6.3 shows the work group "MSHOME" and servers in PLANT_SERVER.

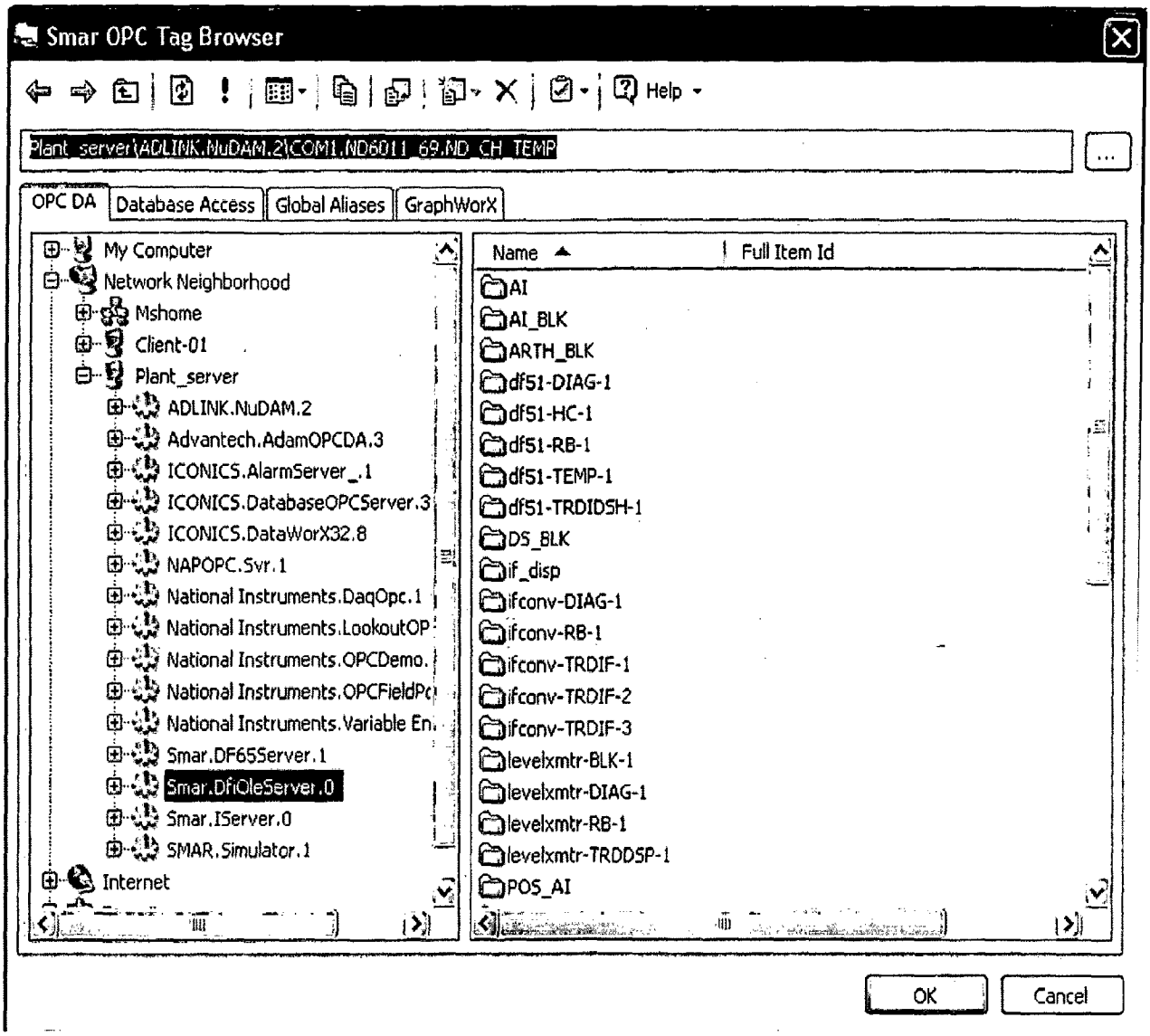


Fig 6.3 Smar OPC Tag browser

The figures 6.4 to 6.12 show the front panels developed in GraphWorx and LabVIEW in context of the model Hydro Power station and that of a test OPC client developed in Visual basic displaying real-time values of the parameters. The Web pages viewed through internet explorer are also attached.

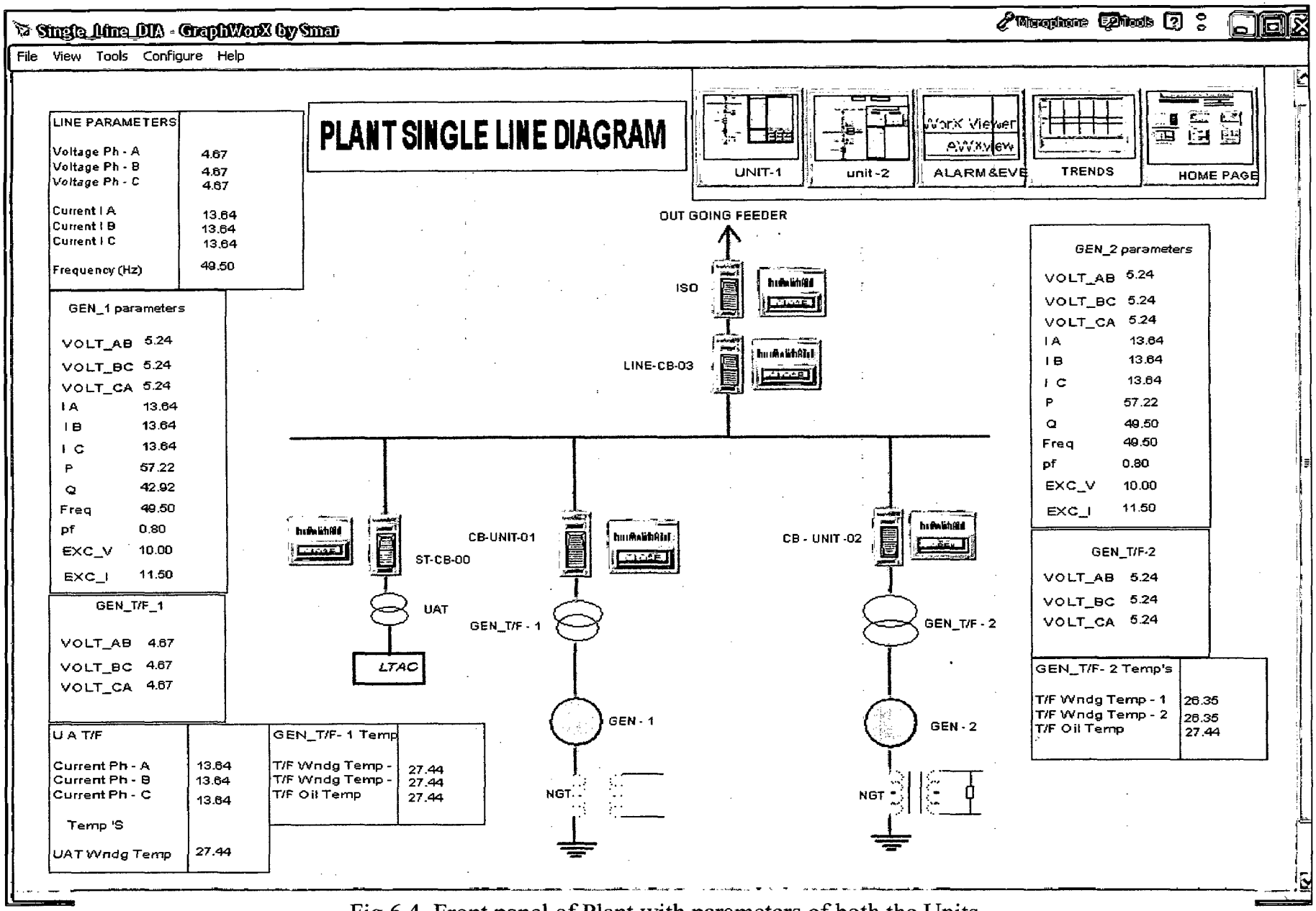


Fig.6.4. Front panel of Plant with parameters of both the Units

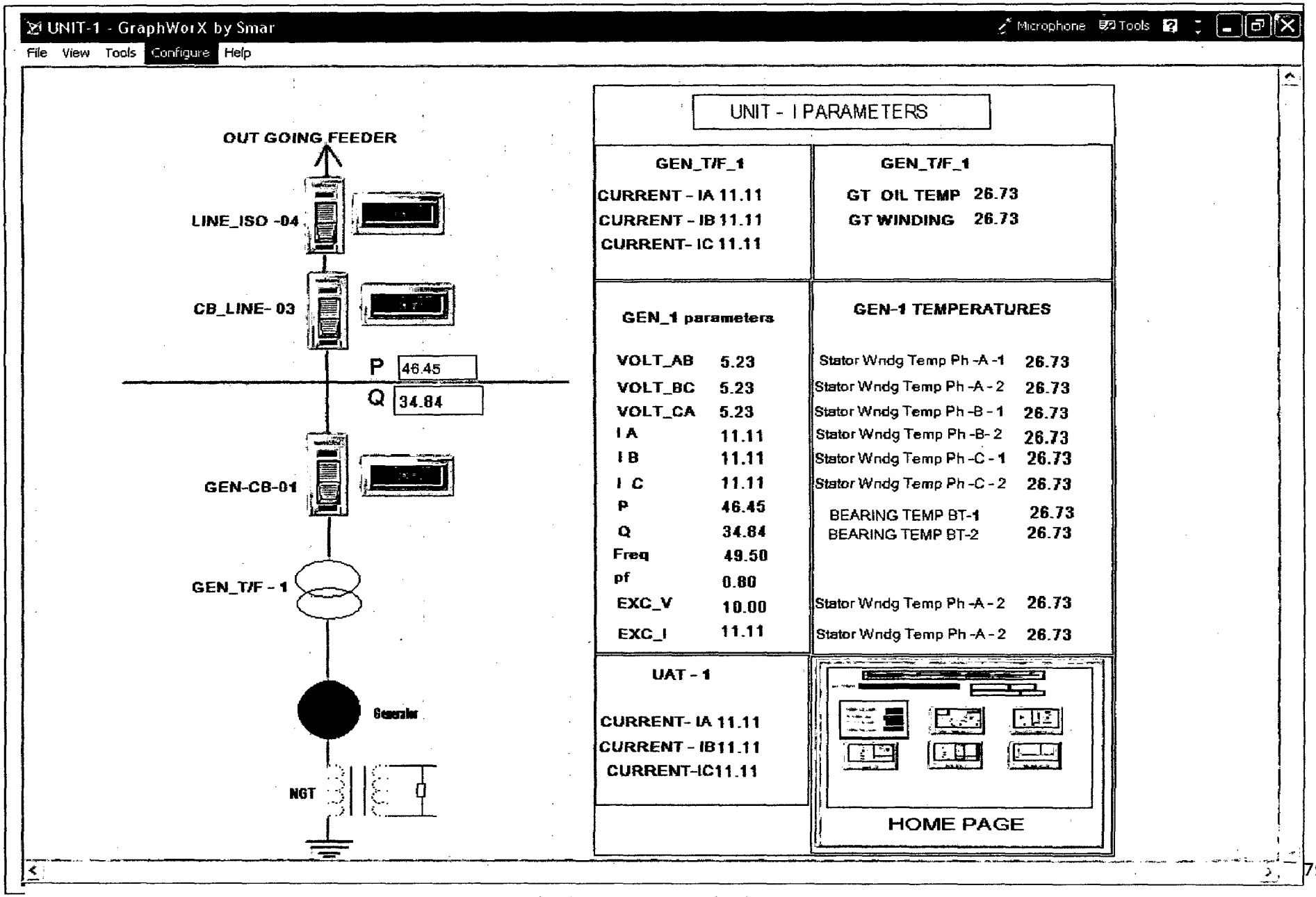


Fig.6.5. Front Panel of UNIT - 1

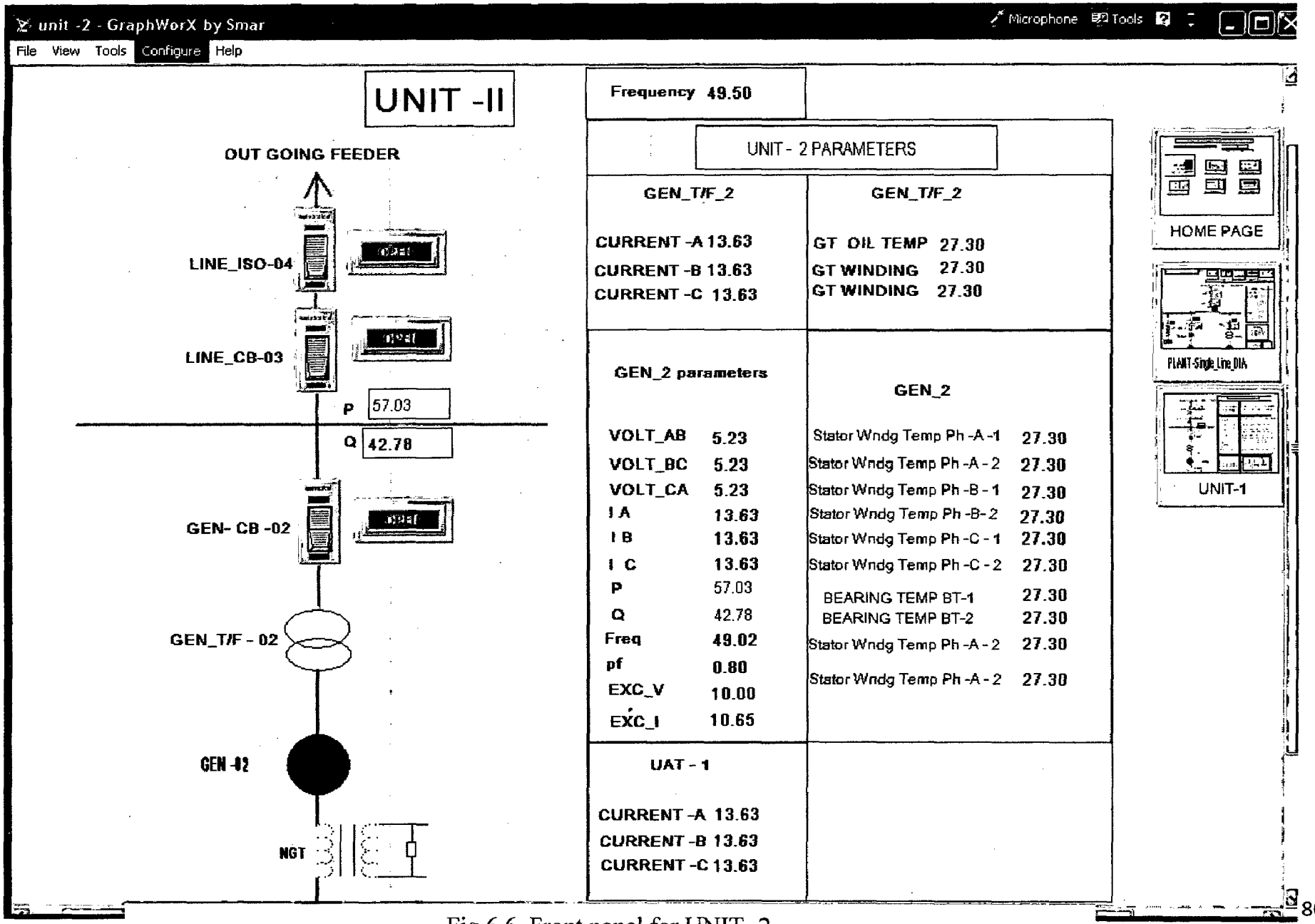


Fig.6.6. Front panel for UNIT -2

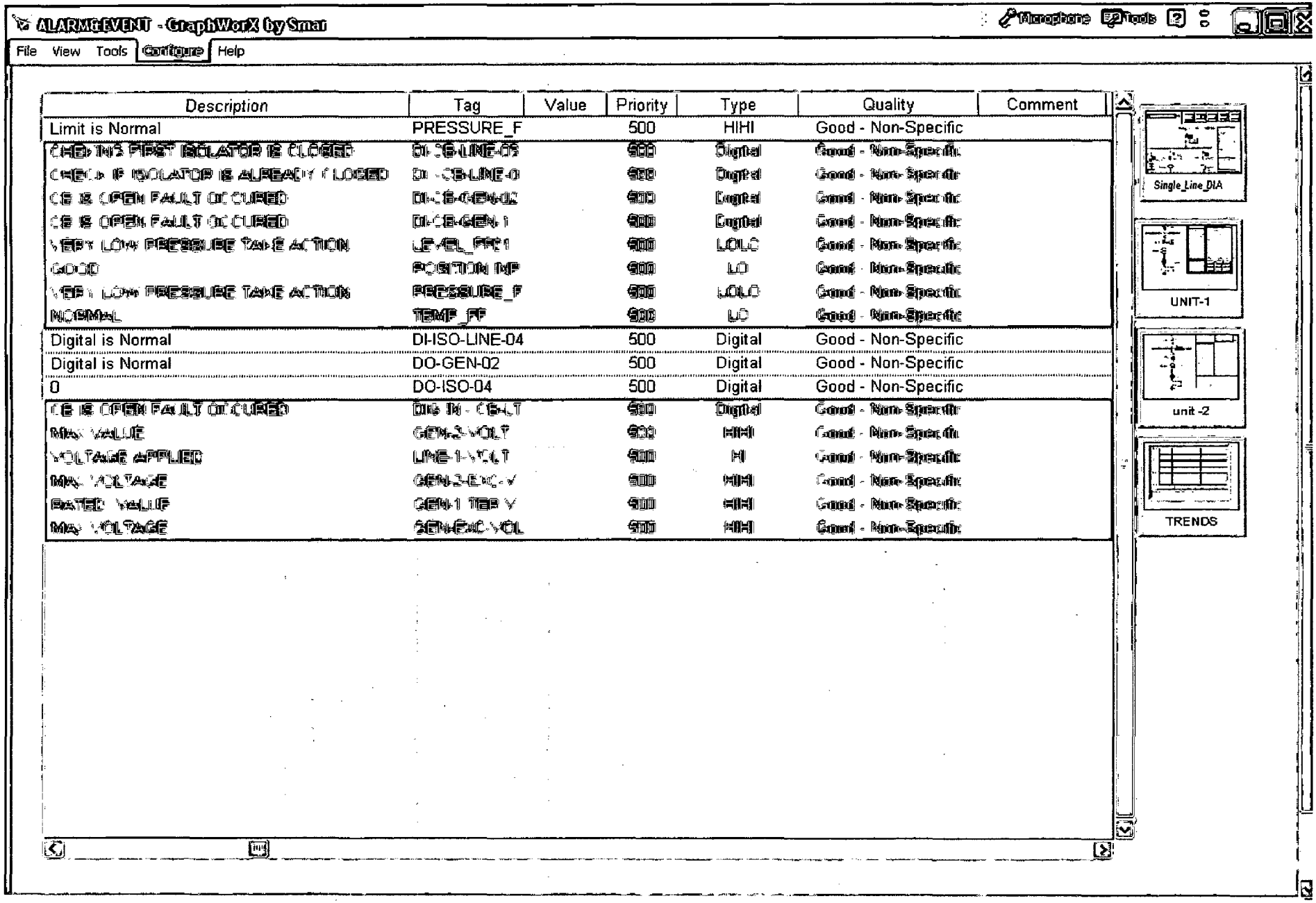


Fig.6.7. Alarm WorX Viewer

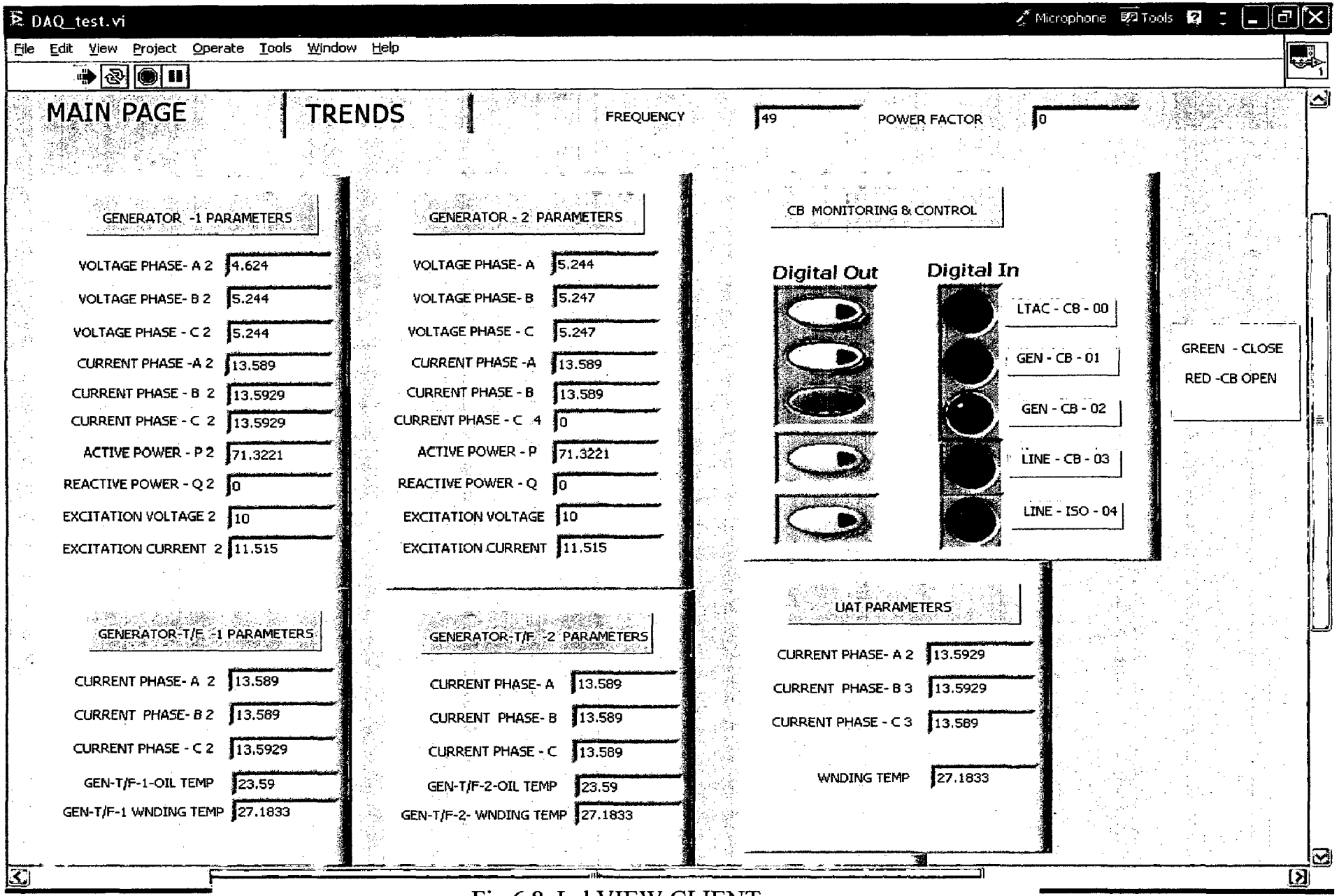


Fig.6.8. LabVIEW CLIENT

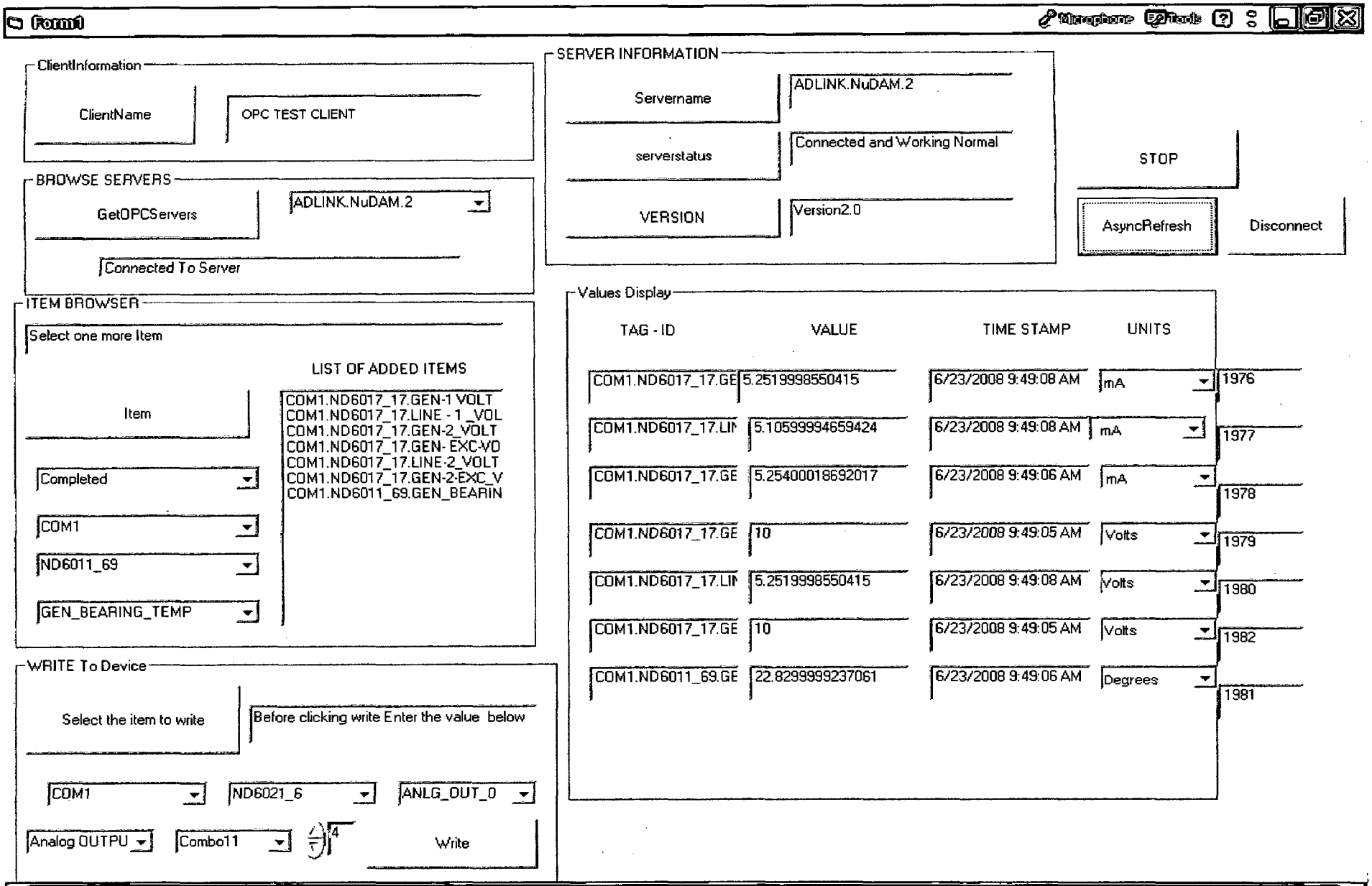


Fig.6.9. OPC Test CLIENT in Visual Basic 6.0

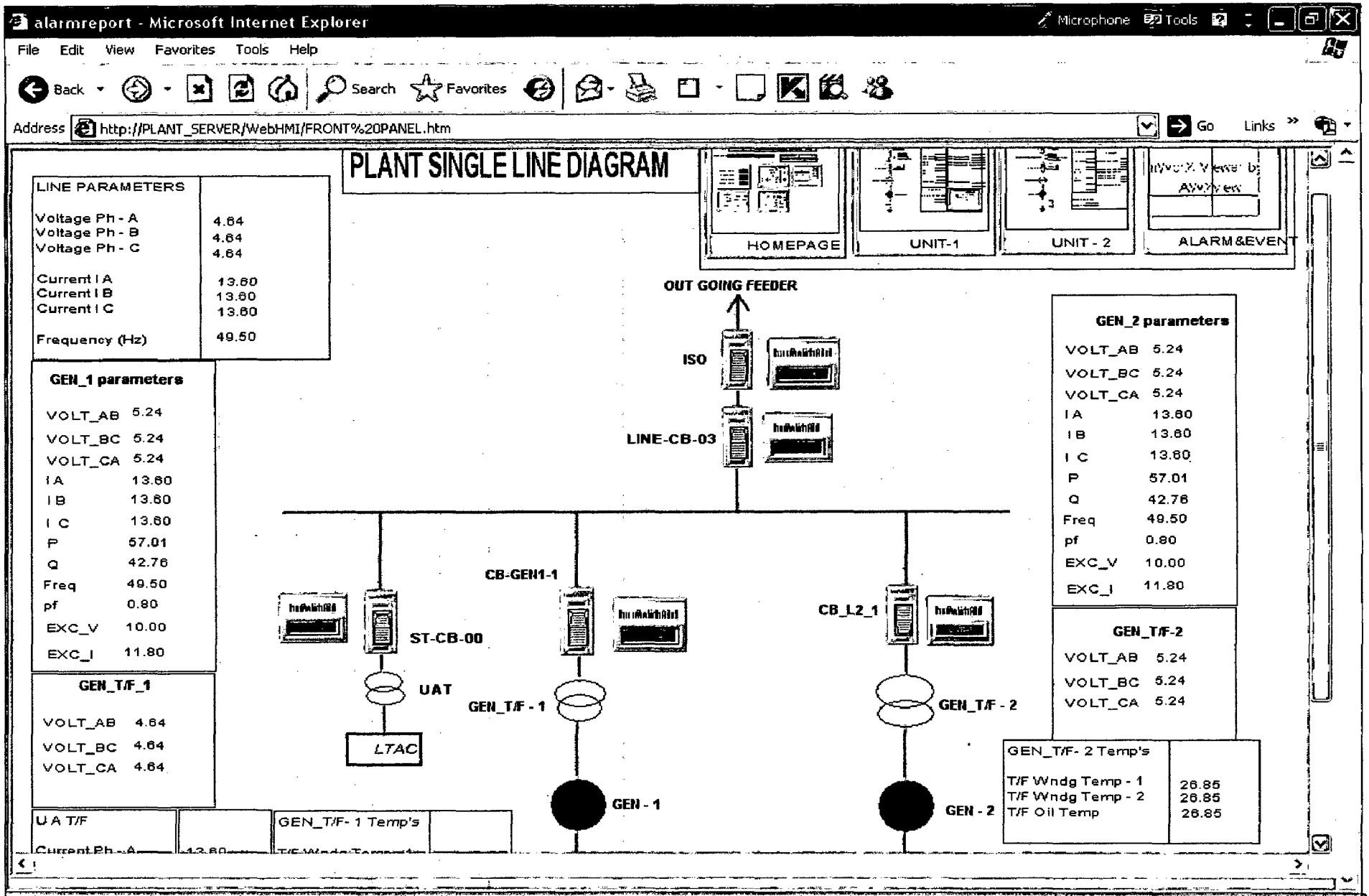


Fig.6.10. Web page of Plant single Line Diagram of 6.4

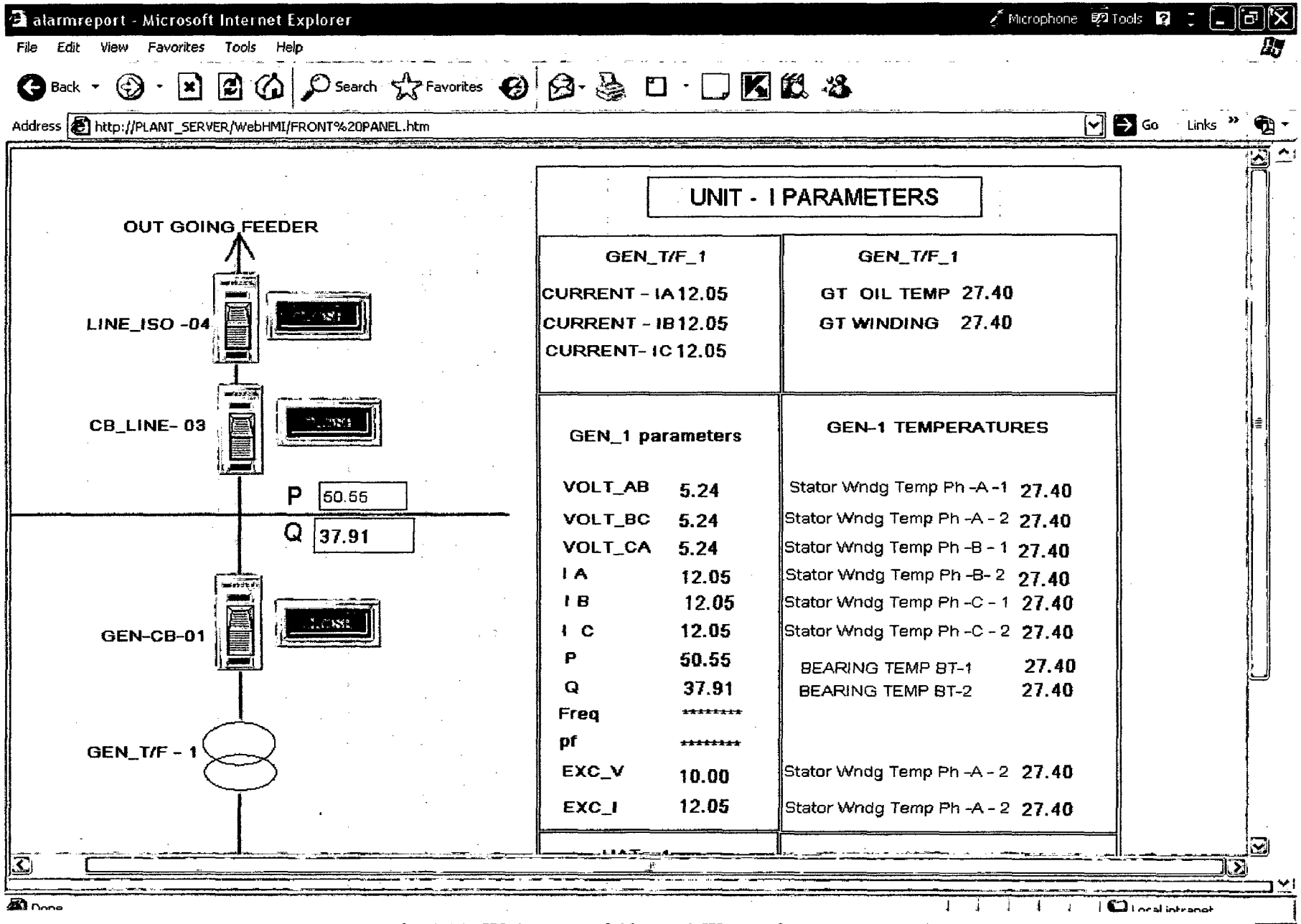


Fig.6.11. Web page of Plant UNIT -I of 6.4

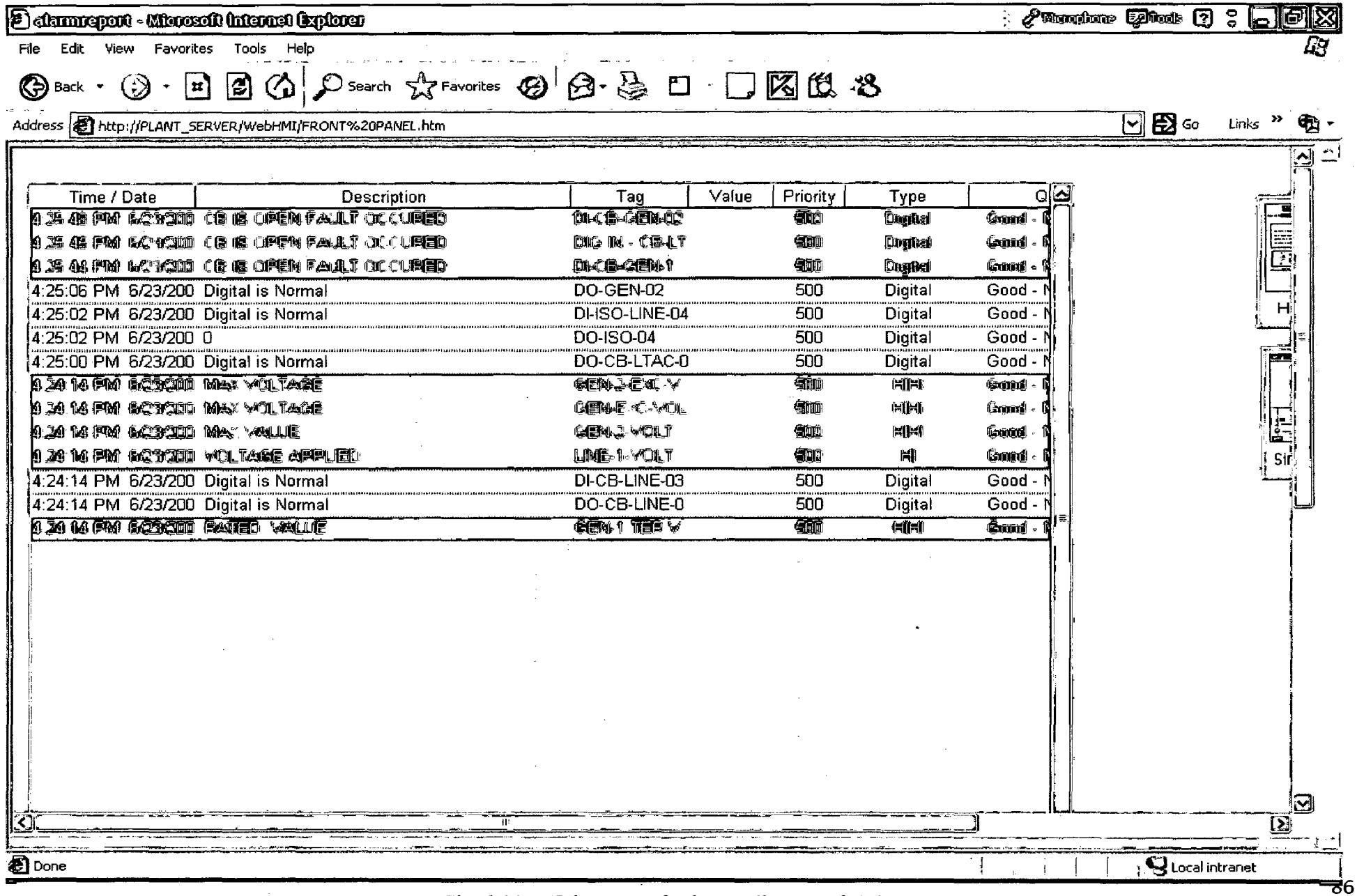


Fig.6.12. Web page of Alarm Viewer of 6.6

CHAPTER 7 Conclusions & Future Scope

In this chapter the various concluding remarks and the future scope of the work done are summarized.

7.1 Conclusions

The Mixed-mode smart sensor network with two remote clients, integrating a FOUNDATION FIELDBUS and two MODBUS networks is successfully developed with available smart and Intelligent sensors in the laboratory. The two remote clients were also successfully communicated with three servers on the central server node. The developed mixed-mode network is realized as an application to a SCADA network of a model Hydro Power station. The functions namely, real-time data acquisition and display on the remote nodes in the front panels developed using multiple client software, control functions are also simulated in the laboratory, alarm and event notifications through alarm servers for the operator on remote clients, front panels so developed are published as web pages to a web server installed on the central server node so that the nodes without the client software components can also view the real-time parameters through a web browser, were successfully implemented. The OPC security is also employed by specifying the user groups for access, the specific users were also restricted for access to only few applications like only monitoring but no controlling rights. Hence the OPC Data access, OPC Alarm and Event, OPC XML data access and OPC security standards were successfully implemented on a local area network.

7.2 Future Scope

The mixed-mode SCADA network developed is functioning well and this can implemented in any Small Hydro power station or on a setup. Although few control functions are implemented with available equipment, complex control strategies implemented in industrial control systems can be programmed using FIELDBUS function blocks, demonstrating a fully distributed network.

References

[1] Zaiping Chen, Xiaowei Yao, Xunlei Yin "Research of Schemes on Integration of Fieldbus System", Industrial Electronics Society, 31st Annual Conference of IEEE Nov- 2005,pp 6-10.

[2] Maxim Lobashov1, Thilo Sauter2, "Vertical Communication from the Enterprise Level to the Factory Floor – Integrating Fieldbus and IP-based Networks", Emerging Technologies and Factory Automation, ETFA '06. IEEE Conference on September 2006, pp-20-22.

[3] Li Zheng, Nakagawa. H, "OPC (OLE for process control) specification and its developments", Proceedings of the 41st SICE Annual Conference, ,vol.2, Aug. 2002, pp:917 - 920

[4] Vu Van Tan, Dae-Seung Yoo, Myeong-Jae Yi, "Design and Implementation of Web Service by Using OPC XML-DA and OPC Complex Data for Automation and Control Systems", Computer and Information Technology, the Sixth IEEE International Conference , Sept. 2006 pp:263 – 263

[5] Xiaohong Hao, Shunhong Hou, "OPC DX and industrial Ethernet glues fieldbus together", Control, Automation, Robotics and Vision Conference-8th, Volume 1, 6-9 Dec. 2004 Page(s):562 - 567

[6] What is OPC at www.opcfoundation.org http://www.opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC

[7] OPC Common Definitions and Interfaces, Version 1.0, October 27, 1998, at www.opcfoundation.org

[8] Data Access Automation Interface Standard, Version 2.02, February 4, 1999, at www.opcfoundation.org

[9] Data Access Custom Interface Standard, Version 2.0, October 14, 1998, at www.opcfoundation.org

[10] OPC Net wrapper- Client interface manual, Version 1.1.0.0, 22 March 2005, at www.opcfoundation.org

[11] Performance Test Report of ALEO SHP Station (2 * 1500 kW), AHEC, IIT Roorkee , Dec 2005.

[12] Performance Test Report and SCADA manuals of Someshwara SHP Station, AHEC, IIT Rorkee, 2005.

[13] NAPOPC DA Server, User's Manual [For Windows 95/98/Me/NT/2000/XP], Version: 3.00, Date: Jul-10-2007

[14] Smar OLE Server, User manual, July 2002, VERSION 2.0

[15] NDS-OPC OPC Server for NuDAM Modules, Windows-95/98/NT, Manual Rev. 1.00b: September 6, 1999

[16] "Fieldbus Tutorial", Smar International Corporation, USA, July 2004.

[17] "Operation and Instruction/ Manual model LD 302", Smar International Corporation, USA, Dec 2005.

[18] "Operation and Instruction/ Manual model TP 302", Smar International Corporation, USA, July 2006.

[19] "Operation and Instruction/ Manual model IF 302", Smar International Corporation, USA, Nov 2006.

[20] "Operation and Instruction/ Manual model TT 302", Smar International Corporation, USA, Feb 2007.

[21] "Operation and Instruction/ Manual model BT 302", Smar International Corporation, USA, March 2005.

[22] "Operation and Instruction/ Manual model DFI 302", Smar International Corporation, USA, July 2005.

[23] SYSCON Manual, Version 6.0, Smar International Corporation, USA, March 2007.

[24] "Foundation Fieldbus General Manual", Smar International Corporation, USA, July 2005.

[25] "Foundation Fieldbus Function Block Manual", Smar International Corporation, USA, May 2007.

[26] Electronic Instruments and Instrumentation Technology – M.M.S.Anand (PHI)

[27] "Operation and Instruction/ Manual Graph WorX", Smar International Corporation, USA, Feb 2007.

[28] "Operation and Instruction/ Manual Smar WebHMI", Smar International Corporation, USA, Feb 2007.

[29] "Operation and Instruction/ Manual WEB Publishing Wizard", Smar International Corporation, USA, Feb 2007.

[30] "Operation and Instruction/ Manual Alarm WorX server Configurator", Smar International Corporation, USA, Feb 2007.

[31] Al Chisholm, "DCOM, OPC and Performance Issues", Intellution Inc, 2/3/98

[32] Karl-Heinz Deiretsbacher, Siemens AG, Jim Luth, ICONICS Inc, OPC Foundation technical Director, Rashesh Mody Invensys/Wonderware OPC Foundation Chief Architect "Using OPC via DCOM with Microsoft Windows XP Service Pack 2".

[33] COM & DDE Technologies at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/callcomcomp.asp

[34] A presentation on "COM, DCOM and OPC" by ICONICS

[35] Guy Eddon and Henry Eddon, "Understanding the DCOM Wire Protocol by Analyzing Network Data Packets", Microsoft Systems Journal, March 1998.

[36] Prof. Dr. H. Kirrmann, ABB Research Centre, Baden, Switzerland " A presentation on OPC",May 2006.

[37] John Weber, President & Founder, Software Toolbox, Inc. "Using Visual Basic As An OPC Client", March 2001

[38] OPC XML-DA Specification, Version 1.0, July 12, 2003

[39] OPC resources at www.ni.com/opc/opc_resources.htm

APPENDIX – A: OPC Data Access Automation Objects and Interfaces

A.1 OPC Server Object

Properties:

StartTime	CurrentTime	LastUpdateTime
MajorVersion	MinorVersion	BuildNumber
VendorInfo	ServerState	LocaleID
Bandwidth	OPCGroups	PublicGroupNames
ServerName	ServerNode	ClientName

Methods:

GetOPCServers	Connect	Disconnect
CreateBrowser	GetErrorString	QueryAvailableLocaleIDs
QueryAvailableProperties	GetItemProperties	LookupItemIDs

Events:

ServerShutDown		

A.2 OPC Browser Object

Properties:

Organization	Filter	DataType
AccessRights	CurrentPosition	Count

Methods:

Item	ShowBranches	ShowLeafs
MoveUp	MoveToRoot	MoveDown
MoveTo	GetItemID	GetAccessPaths

A.3 OPC Groups Object

Properties:

Parent	DefaultGroupIsActive	DefaultGroupUpdateRate
DefaultGroupDeadband	DefaultGroupLocaleID	DefaultGroupTimeBias
Count		

Methods:

Item	ShowBranches	ShowLeafs
MoveUp	MoveToRoot	MoveDown
MoveTo	GetItemID	GetAccessPaths

Events:

GlobalDataChange		

A.4 OPC Group Object

Properties

Parent	Name	IsPublic
IsActive	IsSubscribed	ClientHandle
ServerHandle	LocaleID	TimeBias
DeadBand	UpdateRate	OPCItems

Methods

SyncRead	SyncWrite	AsyncRead
AsyncWrite	AsyncRefresh	AsyncCancel

Events

DataChange	AsyncReadComplete	AsyncWriteComplete
AsyncCancelComplete		

A.5 OPC Items Object

Properties

Parent	DefaultRequestedDataType	DefaultAccessPath
DefaultIsActive	Count	

Methods

Item	GetOPCItem	AddItem
AddItems	Remove	Validate
SetActive	SetClientHandles	SetDataTypes

A.6 OPC Item Object

Properties

Parent	ClientHandle	ServerHandle
AccessPath	AccessRights	ItemID
IsActive	RequestedDataType	Value
Quality	TimeStamp	CanonicalDataType
EUType	EUInfo	

Methods

Read	Write	

```
End Sub
-----------------------------------------------------------------------
Private Sub Combo2_Click()

AnBrowser.MoveDown (Combo2.Text)  /*Moving to the selected position
AnBrowser.ShowBranches                in the tree structure of server*/
Dim i As Integer

For i = 1 To AnBrowser.Count
 Combo3.AddItem (AnBrowser.Item(i))
 Next i

End Sub
-----------------------------------------------------------------------
Private Sub Combo3_Click()

Dim i As Integer
AnBrowser.MoveDown (Combo3.Text)
AnBrowser.ShowLeafs              /* Accessing all the items in server*/
Combo7.Clear
   For i = 1 To AnBrowser.Count
     Combo7.AddItem (AnBrowser.Item(i))
   Next i

End Sub
-----------------------------------------------------------------------
Private Sub Combo4_Click()

Dim i As Integer
AnBrowser.MoveDown (Combo4.Text)
AnBrowser.ShowBranches
Combo5.Clear
    For i = 1 To AnBrowser.Count
      groups1(i) = AnBrowser.Item(i)
      Combo5.AddItem groups1(i)
    Next i

End Sub
-----------------------------------------------------------------------
Private Sub Combo5_Click()

Dim othername As String
Dim i As Integer
AnBrowser.MoveDown (Combo5.Text)
AnBrowser.ShowLeafs
Combo6.Clear
For i = 1 To AnBrowser.Count
othername = AnBrowser.Item(i)
Combo6.AddItem othername
Next i

End Sub
-----------------------------------------------------------------------
Private Sub Combo6_Click()

OPCItemIDs(k) = AnBrowser.GetItemID(Combo6.Text)
```

Events:

GlobalDataChange		

A.4 OPC Group Object

Properties

Parent	Name	IsPublic
IsActive	IsSubscribed	ClientHandle
ServerHandle	LocaleID	TimeBias
DeadBand	UpdateRate	OPCItems

Methods

SyncRead	SyncWrite	AsyncRead
AsyncWrite	AsyncRefresh	AsyncCancel

Events

DataChange	AsyncReadComplete	AsyncWriteComplete
AsyncCancelComplete		

A.5 OPC Items Object

Properties

Parent	DefaultRequestedDataType	DefaultAccessPath
DefaultIsActive	Count	

Methods

Item	GetOPCItem	AddItem
AddItems	Remove	Validate
SetActive	SetClientHandles	SetDataTypes

A.6 OPC Item Object

Properties

Parent	ClientHandle	ServerHandle
AccessPath	AccessRights	ItemID
IsActive	RequestedDataType	Value
Quality	TimeStamp	CanonicalDataType
EUType	EUInfo	

Methods

Read	Write	

APPENDIX – B: Program for OPC test client in visual Basic

```
Option Explicit
Option Base 1 'Makes all arrays start with an index of 1
-------------------------------------------------------------
Dim WithEvents ConnectedOPCServer As OPCServer
Dim WithEvents MyGroups As OPCGroups
Dim WithEvents oneGroup As OPCGroup
Dim Groupname As String
Dim AnBrowser As OPCBrowser
Dim k As Integer                    /* Intialising the objects and
Dim Myitems As OPCItems                variables */
Dim oneItem As OPCItem

Dim ItemCount As Long
Dim OPCItemIDs(30) As String
Dim groups1(30) As Variant
Dim groups2(30) As Variant
Dim ItemServerHandles(50) As Long
Dim ItemServerErrors(50) As Long
Dim ClientHandles(50) As Long
-------------------------------------------------------------
Private Sub AsyncRefresh_Click()

Dim Source As Integer
Dim ClientTransactionID As Long        /*This Function Fires the
Dim ServerTransactionID As Long         OnDatachange Event for all
Source = 1                                    Added groups */
ClientTransactionID = 2125
oneGroup.AsyncRefresh Source, ClientTransactionID, ServerTransactionID

End Sub
-------------------------------------------------------------------
Private Sub Client_Name_Click()
Dim info As String
info = ConnectedOPCServer.ClientName     /* Accessing the Client name
Text3.Text = info                           Property of server Object*/
End Sub
-------------------------------------------------------------------
Private Sub Combo1_Click()

ConnectedOPCServer.Connect (Combo1.Text)   /*Connecting the server
List1.AddItem "Connected To Server"          to the selected server*/
Set MyGroups = ConnectedOPCServer.OPCGroups /*Getting the reference of
Set oneGroup = MyGroups.Add("Group_Read")     OPCgroups from server */

Set AnBrowser = ConnectedOPCServer.CreateBrowser /* Getting the
            reference of the Browser object from the connected server*/

oneGroup.UpdateRate = 500
oneGroup.IsActive = True
oneGroup.IsSubscribed = True
k = 1
ItemCount = 25
ConnectedOPCServer.ClientName = "OPC TEST CLIENT "
```

```
End Sub
-----------------------------------------------------------------------
Private Sub Combo2_Click()

AnBrowser.MoveDown (Combo2.Text)  /*Moving to the selected position
AnBrowser.ShowBranches                in the tree structure of server*/
Dim i As Integer

For i = 1 To AnBrowser.Count
 Combo3.AddItem (AnBrowser.Item(i))
 Next i

End Sub
-----------------------------------------------------------------------
Private Sub Combo3_Click()

Dim i As Integer
AnBrowser.MoveDown (Combo3.Text)
AnBrowser.ShowLeafs              /* Accessing all the items in server*/
Combo7.Clear
   For i = 1 To AnBrowser.Count
     Combo7.AddItem (AnBrowser.Item(i))
   Next i

End Sub
-----------------------------------------------------------------------
Private Sub Combo4_Click()

Dim i As Integer
AnBrowser.MoveDown (Combo4.Text)
AnBrowser.ShowBranches
Combo5.Clear
    For i = 1 To AnBrowser.Count
      groups1(i) = AnBrowser.Item(i)
      Combo5.AddItem groups1(i)
    Next i

End Sub
-----------------------------------------------------------------------
Private Sub Combo5_Click()

Dim othername As String
Dim i As Integer
AnBrowser.MoveDown (Combo5.Text)
AnBrowser.ShowLeafs
Combo6.Clear
For i = 1 To AnBrowser.Count
othername = AnBrowser.Item(i)
Combo6.AddItem othername
Next i

End Sub
-----------------------------------------------------------------------
Private Sub Combo6_Click()

OPCItemIDs(k) = AnBrowser.GetItemID(Combo6.Text)
```

```
List4.AddItem OPCItemIDs(k)
AnBrowser.ShowLeafs
Set Myitems = oneGroup.OPCItems
Dim x, i As Integer
x = 1975
i = 1
 ClientHandles(k) = x + I /* Setting client Handles for the items
                                    to be added*/
   i = i + 1

      Myitems.AddItem OPCItemIDs(k), ClientHandles(k) /* Adding the
                                   Items using "Add Item property"*/
      Set oneItem = Myitems.Item(k)
      Dim somevalue As Long
      ItemServerHandles(k) = oneItem.ServerHandle /* Getting the Server
                                        Handles of the added Items*/

AnBrowser.MoveToRoot
k = k + 1
If k < ItemCount Then
Text1.Text = "Select one more Item "
Else
Text1.Text = " Items selection is completed "
End If

End Sub
-------------------------------------------------------------------------
Private Sub Command1_Click()
End                        /* End of Execution*/
End Sub
-------------------------------------------------------------------------
Private Sub Disconnect_Click()

MyGroups.RemoveAll   /*Removes all the added items and groups */
ConnectedOPCServer.Disconnect
End Sub

-------------------------------------------------------------------------
Private Sub Form_Load()

Combo8.AddItem "Completed"
Combo8.AddItem "NotCompleted"

End Sub
-------------------------------------------------------------------------
Private Sub Getopcservers_Click()

Dim Allopcservers As Variant
Dim i As Integer
Set ConnectedOPCServer = New OPCServer   /*Creating a dummy server
                                                    object/*
Allopcservers = ConnectedOPCServer.Getopcservers /*Getting all the
                               registered OPC server in the node*/

For i = LBound(Allopcservers) To UBound(Allopcservers)
Combo1.AddItem Allopcservers(i)
Next i
```

```
End Sub
-----------------------------------------------------------------------
Private Sub Item_Click()

Dim i As Integer
Dim somename As String
Dim currentvalue As Long


Set MyGroups = ConnectedOPCServer.OPCGroups
MyGroups.DefaultGroupIsActive = True
MyGroups.DefaultGroupDeadband = 0     /* Setting the properties*/

Dim position As String
AnBrowser.ShowBranches
Combo4.Clear
For i = 1 To AnBrowser.Count

somename = AnBrowser.Item(i)
Combo4.AddItem somename
Next i

End Sub
-----------------------------------------------------------------------
Private Sub oneGroup_DataChange(ByVal TransactionID As Long, ByVal
NumItems As Long, ClientHandles() As Long, ItemValues() As Variant,
Qualities() As Long, TimeStamps() As Date)

Dim i As Integer

If Combo8.Text = "Completed" Then
List6.Clear

For i = 1 To k - 1
List6.AddItem OPCItemIDs(i) & "              " & ItemValues(i) & "
" & TimeStamps(i)
List6.AddItem "                                                    "
Next i              /** This is an Event driven function which will
                        Executed when
Else
End If
End Sub
-----------------------------------------------------------------------

Private Sub Select_Click()

Dim i As Integer
Dim somename As String

AnBrowser.MoveToRoot
AnBrowser.AccessRights = 4
AnBrowser.ShowBranches
Combo2.Clear
For i = 1 To AnBrowser.Count
somename = AnBrowser.Item(i)
Combo2.AddItem somename
Next i
```

```
'i = AnBrowser.AccessRights
'Text5.Text = i
End Sub
-------------------------------------------------------------------------
Private Sub Servername_Click()
Dim info As String
Dim node As String

info = ConnectedOPCServer.Servername /*Accesing the Server ID*/
List2.AddItem info
node = ConnectedOPCServer.ServerNode
List2.AddItem node

End Sub
-------------------------------------------------------------------------
Private Sub serverstatus_Click()

Dim serverstatus As Long
serverstatus = ConnectedOPCServer.ServerState
If serverstatus = 1 Then  /* Readin the Current status of the Server*/
List3.AddItem "Connected and Working Normal"
End If
End Sub
-------------------------------------------------------------------------
Private Sub Version_Click()

Dim major, minor As Integer
major = ConnectedOPCServer.MajorVersion /*Reading version of the
minor = ConnectedOPCServer.MinorVersion      Connected server*/
Text4.Text = "Version" & major & "." & minor
End Sub
-------------------------------------------------------------------------
Private Sub Write_Click()

Dim oneGroup1 As OPCGroup
Dim Myitems1 As OPCItems
Dim oneItem1 As OPCItem

Dim Value As Variant
Dim ClientHandle As Long

Set oneGroup1 = MyGroups.Add("Group_Write")/*Adding the group to
oneGroup1.IsActive = True                          write*/
oneGroup1.IsSubscribed = True
Set Myitems1 = oneGroup.OPCItems

ClientHandle = 2000   /*setting the client Handle to added item */
Myitems1.AddItem AnBrowser.GetItemID(Combo7.Text), ClientHandle
Set oneItem1 = Myitems1.Item(1)
'Text2.Text = "Enter the Value to be written"

Value = Text2.Value
oneItem1.Write (Value)
'List5.AddItem oneItem1

End Sub
-------------------------------------------------------------------------
```