ALICE / 99-14

Internal note-DCS

First experience with PLC, OPC and BridgeVIEW in the context of the HMPID liquid distribution prototype

G. Lecoeur *CERN EP-TA2, Geneva, Switzerland*
E. Määtta *CERN IT-CO, Geneva, Switzerland/Kayaani Polytechnic, Kayaani, Finland*
H. Milcent *CERN IT-CO, Geneva, Switzerland*
D. Swoboda *CERN EST-LEA, Geneva, Switzerland*

Abstract

The TESt (Test and Evaluation Station)[3] project consists in the construction of a stand alone unit for a specific sub-system of an ALICE detector in order to gain first experience with commercial products for detector control. Although the control system includes only a small number of devices and is designed for a particular application, it covers nevertheless all layers of a complete system and can be extended or used in different applications.

The control system prototype has been implemented for the Perfluorohexane (C6F14) liquid distribution of the ALICE-HMPID (High-Momentum Particle IDentification)[5].

This report presents the experience acquired while developing the control system with off the shelf items: National Instruments BridgeVIEW[6] supervision software, SIEMENS PLC S7-300[7] and the programming tool STEP 7. The OPC standard (OLE[12] for Process Control)[8] was used for the communication between BridgeVIEW and the PLC.

Keywords: Control system, SCADA, STEP 7, BridgeVIEW, PLC, OPC.

1 Introduction

The JCOP (Joint COntrol Project)[4] has been set up at the beginning of 1998 to look for common solutions for the detector control of the future LHC experiments. Until a complete proposal can be submitted, it will be necessary to implement already now some control equipment for R&D and test beam runs.

The TESt project was setup in ALICE, it includes the hardware and software configuration of a basic control assembly ready to control the operation of the liquid distribution for the ALICE-HMPID[5] prototype with off the shelf components complying with the currently proposed standards (Fieldbuses[9], PLC[10])

The ALICE HMPID (High-Momentum Particle IDentification) detector is a RICH (Ring Imaging CHerenkov) detector. A prototype has been developed by the EP-TA1 group. The IT-CO group has been requested in the framework of the JCOP to participate in the development of the control system of the liquid distribution of this prototype. Although this application is provisionally limited to the detector prototype, the experience gained is valuable for the control of the LHC experiments. It allows to acquire knowledge in

industrial technologies such as PLCs, programming tools and OPC for data exchange with the supervisory software.

The technologies selected will be described in detail and an appreciation of the development will be given.

2 HMPID liquid distribution prototype

The liquid distribution of the prototype (Figure 1) is a simplified version of the system required for the final HMPID detector.

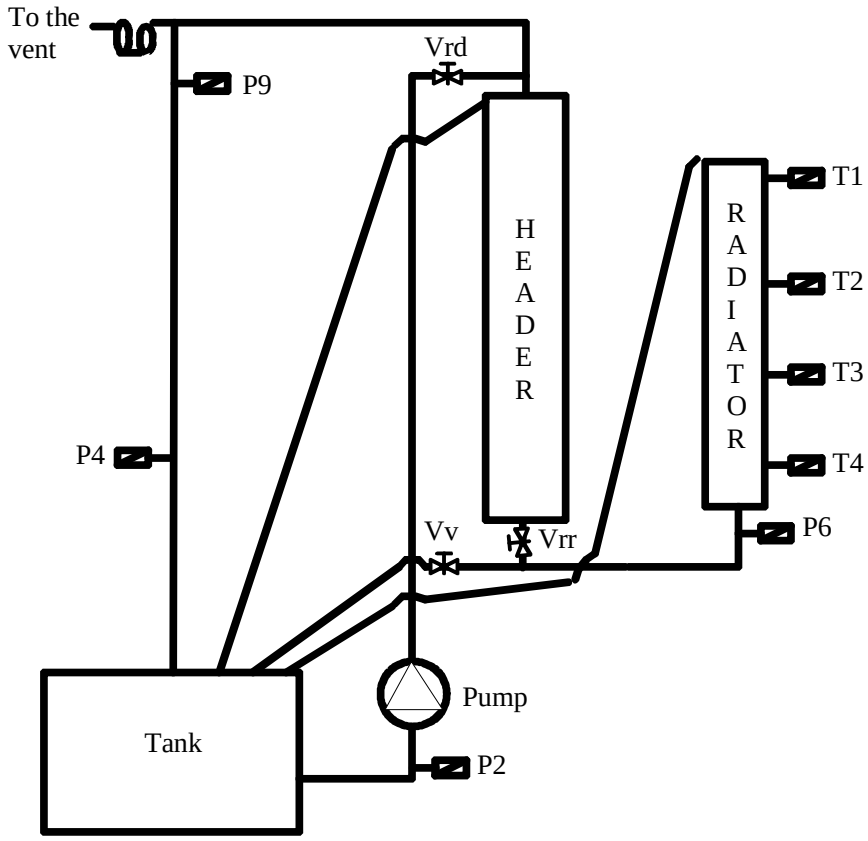

Figure 1: The HMPID C6F14 liquid distribution prototype.

The software architecture for the HMPID prototype includes two distinct layers:

- the process control level at which the actuator and sensor hardware are accessed and the automatic control (closed loop control system) for the liquid distribution is performed
- the supervisory level including Human Machine Interface (HMI), logging, archiving, security and trending.

Both layers can be operated completely independent from each other.

3 Selected technologies

The purpose of the TESt project was to look at off the shelf components to implement this process and supervisory control. The following technologies were used in the HMPID liquid distribution prototype:

- BridgeVIEW for the supervisory layer
- SIEMENS PLC for process control layer
- OPC as a standard interface.

3.1 BridgeVIEW

BridgeVIEW is a SCADA (Supervisory Control And Data Acquisition) software tool that includes real-time process monitoring, alarm handling, event logging, real-time and historical trending, on-line configuration and also control interfaces in form of OPC clients. A graphical programming language is provided for the development of the SCADA application and the HMI (Human Machine Interface) interface.

BridgeVIEW has a tag-based architecture. A tag is a connection to a physical I/O point or a derived parameter and is acquired by Servers. An event is anything that happens to a tag or to the BridgeVIEW engine in general (Figure 2).

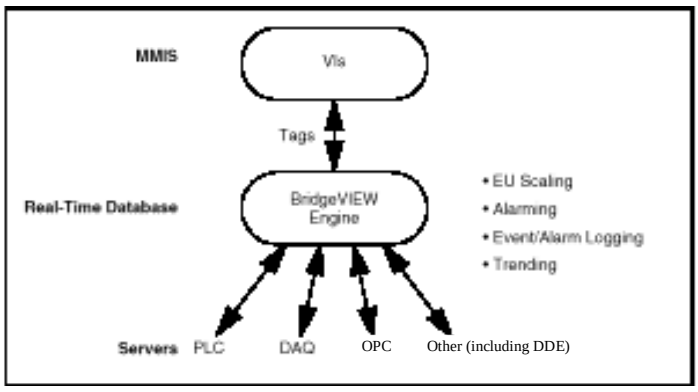

Figure 2: BridgeVIEW architecture.

The BridgeVIEW Engine is the heart of the system. It is the interface between the Servers and the HMI. The Engine exchanges tags with the user process and parameter values with the Servers via input/output queues. It updates a RTDB (Real-Time Data Base) each time a tag value changes by more than its dead-band (Figure 8) and logs possible events (alarm events or operator events). The engine, runs as a separate task completely independent from the HMI application and the Servers.

BridgeVIEW contains a large number of VIs (Virtual Instruments) to implement the registration of the Server and the communication between the Server and the engine. This allows an easy development of VI-based Servers in G code. The VI-based Server and the engine share the same thread in the run-time environment.

3.1.1 Configuration

The information about the servers (OPC, VI-based, etc.) is stored in the Common Configuration Database file (.ccdb) (Figure 3). The tag configuration used by the engine and the user process is kept in the SCADA Configuration File (.scf) and is edited with the tag configurator tool. This information is loaded into the RTDB when the engine is started. Tags are linked to OPC items or to an item declared in the .ccdb file by the Device Servers. Several tags can be linked to the same item. A tag (Figure 4) is assigned to a unique group (Figure 5).

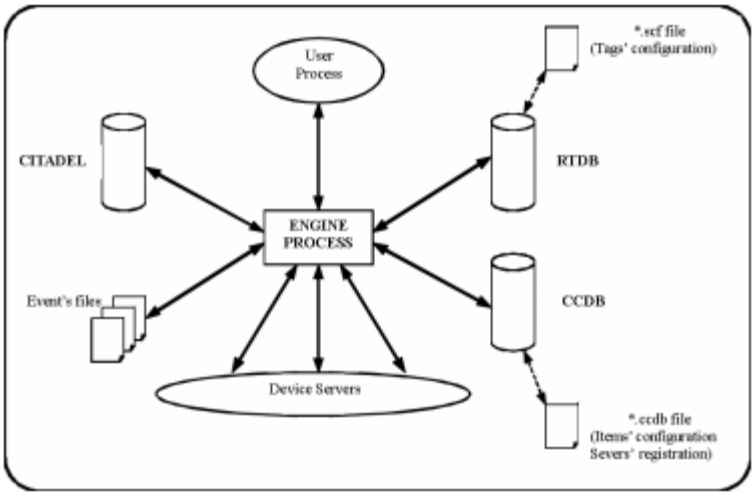

Figure 3: BridgeVIEW configuration environment

A typical development activity with BridgeVIEW includes the:

- declaration and configuration of items, which will be handled by a device server
- declaration and configuration of tags
- configuration / modification of HMI front panel and block diagram.

Figure 4: The BridgeVIEW tag definition panel

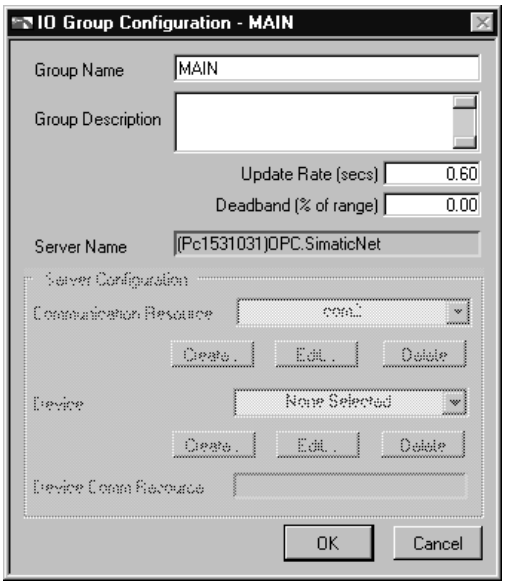

Figure 5: The BridgeVIEW group definition panel

3.1.2 Alarms

The definition of alarms is attached to tags in BridgeVIEW. Four thresholds and a bad state alarm can be defined for every tag. Alarms are generated by the Engine when a tag value is outside the thresholds. Thresholds are absolute values and have a priority level which characterizes the severity of an alarm state.

Alarms are configured with the tag configuration editor (Figure 6)

Figure 6: The alarm configuration editor

An alarm is acknowledged either manually or automatically when the tag value goes back to normal.

VIs to display or acknowledge the alarms (Figure 7) are available in the BridgeVIEW library and can be incorporated in any view. The behavior can be customized.

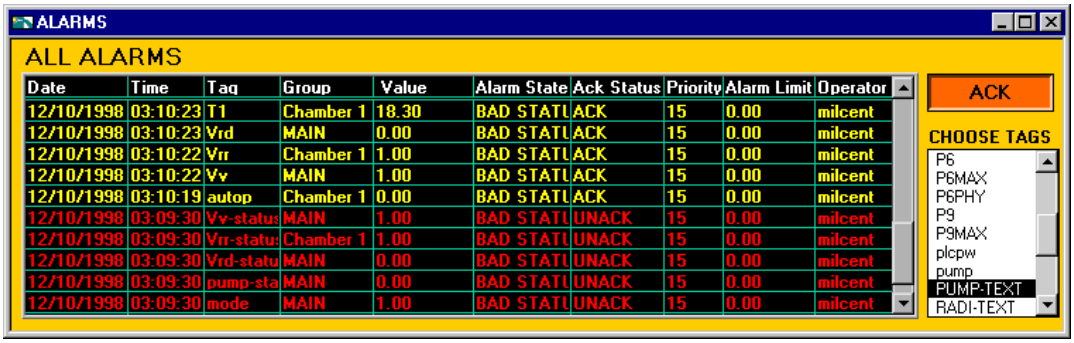

Figure 7: HMPID alarm summary.

3.1.3 Data archiving and logging

A Citadel database is integrated into BridgeVIEW to archive the tag values. A dead-band can be defined for each tag to avoid the archiving of small changes (Figure 8). The archived values can be displayed in the user process views using VIs from the library. They can also be accessed by external applications using the available ODBC driver.

Events can also be archived in dedicated files. An event is generated when the status of an alarm changes. An event can also be generated when the user process overwrites a tag value in case this option has been selected. Events are archived if the source tag has been configured accordingly.

Logged events can be displayed in user process views with the Event History VI from the library. This VI is similar to the Alarm summary VI.

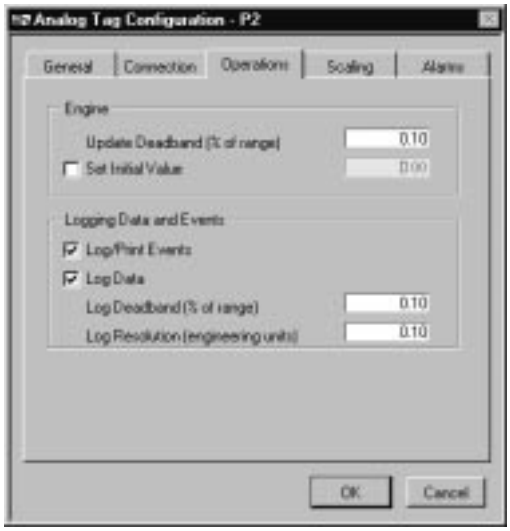

Figure 8: Engine, Archiving and logging setting panel.

Archiving and logging can be globally controlled from a VI or by setting start-up preferences from the Engine Manager.

3.1.4 Access control

BridgeVIEW allows to protect the operation environment and the process (launching of the Engine, start and stop of archiving, etc.). The protection is based on user privileges which are granted by the system administrator to each user.

The process is protected by restricting operation and visibility of HMI items. Administrators may define levels (or groups of users) and give each user a level. A level has a value and a name. The visibility and / or the operation of any HMI item can be restricted by specifying in the G code the minimum level value required to view and / or operate the item. Users of the application have to log in when access control is used.

3.1.5 HMI

HMI in BridgeVIEW consists of two parts, the front panel and the block diagram.

The front panel contains the graphical views. The BridgeVIEW objects library includes a large selection of objects to construct the views (VI). Graphics, Real-time and historical trends can also be integrated in the front panel.

A block diagram with the necessary G code to control the behavior is required for each control panel. A rather large library of VIs covering HMI, mathematics, data analysis, networking, on-line configuration facilities is provided with the G language development environment.The HMI wizard allows to generate G code automatically. This is provided for most graphical items and creates an item specific behavior. The automatically generated code can subsequently be customized by the developer.

3.2 PLC

An increasing number of industrial system is now implementing PLCs (Programmable Logic Controller) for process control. PLCs are compact computers including all the necessary hardware interfaces to the process level. They are generally used for automatic control applications (closed loop control, etc.) either stand-alone or networked through fieldbuses or most recently on industrial ethernet connections.

A PLC includes a power supply, a CPU and input/output modules mounted in a rack. Modules and CPU can be connected through a SIEMENS proprietary backplane bus (Figure 9) or distributed by using a fieldbus such as PROFIBUS[11].

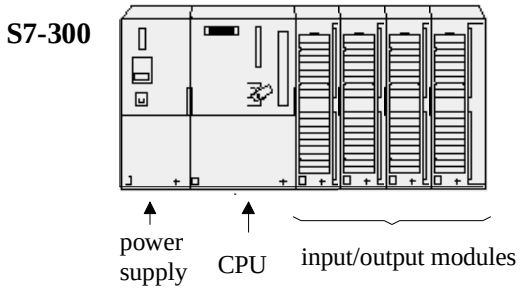

Figure 9: PLC with local input/output module.

The control program for the PLC is set up in a special development environment and then downloaded to the CPU. A user program is composed of the following items:

- OB: Organization Block. OBs are called by the operating system. Three kinds of OBs exist: OBs executed at the start-up of the CPU; OBs executed under defined conditions or at periodic intervals, OBs executed on interrupts or errors detected by the CPU. The available OBs are a function of the CPU type. Each OB has a priority and the highest priority OB is executed before the lowest priority OB. Table 1 shows the different OBs.
- SFC: System Function. This is a function like a system call in the C programming language.
- FC: User Function. It consists of a user supplied function with inputs and outputs but without memory block.
- FB: Function Block. This is similar to a FC with a block of memory (DI) associated to it.
- SFB: System Function Block.
- DB: Data Block. This is a memory area containing data of the user program. DBs can be opened and closed dynamically.
- DI: Instance Data Block. This is a DB associated with a FB. One FB can have multiple DIs.

Table 1: Types of OBs and its priority.

Type of interrupt (CPU clock, hardware/software interrupts, etc.)	OB	Priority
main program scan	OB1	1
time-of-day interrupt	OB10 to OB17	2
time-delay interrupt	OB20 to OB23	3 to 6
cyclic interrupt	OB30 to OB38	7 to 15
hardware interrupt	OB40 to OB47	16 to 23
multicomputing interrupt	OB60	25
asynchronous error interrupts	OB80 to OB87	26
background cycle	OB90	0.29
synchronous error interrupts	OB120, OB122	Priority of the OB that caused the error.

The PLC memory is divided in different areas shown in Table 2.

Table 2: Memory area of a PLC.

memory area	description
Process-image input: I	The CPU reads the digital input modules and records in this area the values.
Process-image output: Q	The CPU sends the values stored in this area to the connected digital output modules.
Bit memory: M	Memory storage for interim result used by the user program.
Timer: T	Storage area for timers.
Counter: Z	Storage area for counters.
Data Block: DB	Data Block containing user data for the user program. A DB is like a shared memory.
Data Block: DI	Instance Data Block assigned for FBs (Functional Block) or SFBs (System Function Block)
Local (temporary) data: L	This area contains the temporary data of the block while the block is being executed.
Peripheral input: PI	Direct access to the data of the local or distributed input modules
Peripheral output: PQ	Direct access to the data of the local or distributed output modules

3.2.1 User program

A user program is composed of OBs edited by the user and downloaded in the CPU. The OBs call FBs, FCs, SFCs and SFBs. FBs and FCs can call SFBs, SFCs and other FBs and FCs.

FBs, FCs and OBs can read and write data in the DBs. FBs can read and write data in DIs (Figure 10). The user program has access to the following CPU memory areas: I, Q, PI, PQ, M, T, DB and Z. There is a set of predefined data types including integer, boolean, etc., it is also possible to define new data types that can be used as template for creating DB blocks.

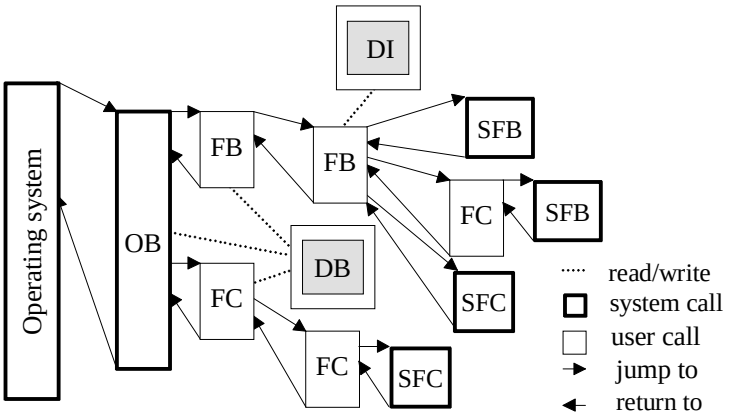

Figure 10: User program module organization.

The number of blocks that can be nested and the number of FCs, FBs and DBs depends on the type of CPU. Figure 11 shows the sequence of a block call within a user program. The instructions of the block are executed when it is called by the program. After the execution of the block, the execution of the previously interrupted block that made the call is resumed at the operation following the block call. Before creating a block, it is necessary to create the variables that are used:

- input and output variables of the block
- static variables which are used in the DI block and preserved after the end of execution of the FB
- dynamic variables which are available during the execution of the block and released when the block is completed.

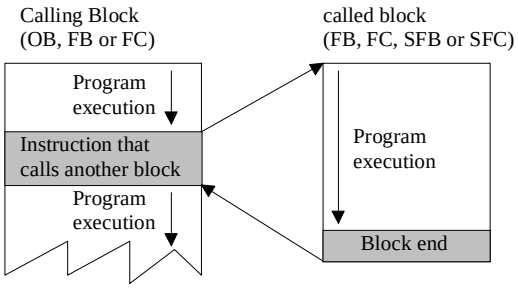

Figure 11: Calling a block.

The SIEMENS PLC operates in polling mode with defined execution cycles. Figure 12 illustrates the phases of the cyclic program execution:

- the operating system starts the cycle monitoring time.
- the operating system copies the value of the digital input module mounted in the CPU rack to the I-memory area of the CPU.
- the operating system processes the user program (OBs) and executes the instructions contained in the user program. The user program has to read / write the analog value of the analog module mounted in the CPU rack and the value of the digital or analog modules distributed via a fieldbus to / from DB- or M- memory area. The user program has read and write access to the I- and Q- memory area.
- the operating system writes the values from the Q- memory area to the digital output modules mounted on the CPU rack.
- at the end of the cycle, the operating system executes any tasks that are pending, for example loading and deleting blocks, receiving or sending data to remote PLC, etc.
- the operating system returns to the start of the cycle and restarts the cycle monitoring task.

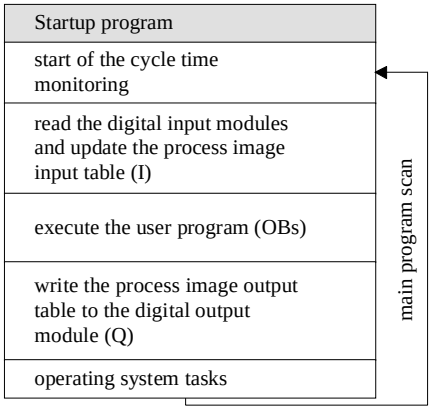

Figure 12: PLC program scan

The cycle time of the CPU is the time to execute the cyclic program and all the program sections resulting from interrupts during the cycle and the time required for system activities. For the S7-300 PLC, OB80 is the block called in the case the cycle time is too small to execute the user program.

The communication load is the load added by the communication to the cycle and impieds on the run time of the user program. Figure 13 illustrates a worst case example with a cycle time set to 1 sec, a user program with a total execution time equal to 750 msec, 250 msec per cycle time for the operating system and a communication load set to 50%. A load set to 50% means that half of each cycle can be used by communication. In this example, the total CPU cycle is tripled. During configuration, the default maximum cycle time and the load can be modified.

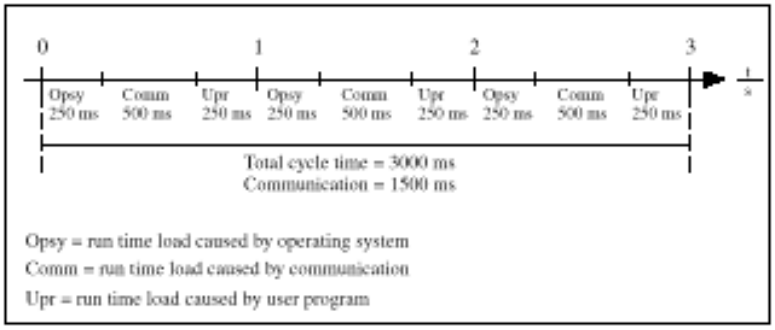

Figure 13: CPU cycle with a 50% load.

The PLC CPU has a keyswitch on its front panel which allows to put the CPU in the states STOP, RUN or RUN-P:

- In the STOP position, all the outputs of the output modules in the same rack are reset.
- The user program is executed when the keyswitch is in RUN or RUN-P position. RUN and RUN-P position are similar except that in RUN-P position, the user can dynamically load or unload OBs, FBs and FCs.

On the transition from STOP to RUN (or RUN-P), the start-up OBs are executed. This is the normal way to set the outputs to a predefined value.

OB80 is the organization block called by the kernel if a call is made to OBs, FBs, or FCs that are not loaded. The kernel puts the CPU in STOP state if the OB80 is missing and a call to a missing block is made. An optional backup battery can be added in order to preserve the PLC configuration and instant data in case of power supply failure.

3.2.2 SIEMENS Communications

Data exchange between SIEMENS PLCs is supported for three different communication systems:

- MPI: MultiPoint Interface, this is a proprietary serial line communication
- PROFIBUS
- Industrial ETHERNET with TCP or ISO protocols.

SIMATIC NET, previously SINEC (SIEMENS Network and Communication) supports the following kinds of communications:

- S7 communications: S7 Functions, a simple and efficient interface between SIEMENS PLCs and PCs over MPI, PROFIBUS or Ethernet.
- SEND / RECEIVE interface: the purpose of the SEND / RECEIVE interface is to link the S7 PLC to the S5 PLC (old version) as well as to other S7 or non S7 stations like a PC. It allows data exchange via PROFIBUS or Ethernet and supports the transfer of medium size of data (up to 240 bytes). A permanent link is established between the PLC and the remote CPU.
- Global Data Communication (GD): facilitates the exchange of cyclic data such as inputs, outputs, memory bits or areas in DBs between CPUs via the MPI interface.

Not all the SIEMENS CPUs support all S7 functions. For example a S7 CPU 3xx can only initiate SEND/RECEIVE connection to the remote partners, but it can receive S7 communication calls. The number of SEND/RECEIVE link and GD (Global Data Communication) depends on the CPU type.

The SIEMENS S7 PLC supports multicomputing that is simultaneous and/or synchronized operation on different CPUs in the same rack. A maximum of four CPUs can be mounted in the same rack.

The SAPI-S7 (Simple Application Programmers Interface) is the S7 functions for the PC to exchange data between a PC and a S7 PLC. It is a connection oriented transport protocol which provides a virtual channel for the user. The communication consists of the following:

- logon
- connection establishment
- data exchange
- connection termination.

In the PC, using the COML S7 tool, a S7 database has to be created where the name of the virtual device (VFD), the address of the device (IP number, Ethernet address or PROFIBUS address) and the name of the connection are defined.

3.2.3 Development tool set environment

STEP 7 is the development software used to configure and program the S7 PLCs. It is a set of tools for:

- setting up and managing a project
- configuring and assigning parameters to hardware
- configuring communication links
- creating the control programs for the PLC
- downloading and testing the programs in the target system.

STEP 7 includes three languages of the IEC 1131-3 (International Electrotechnical Commission's standard) set:

- LADder logic (LAD). The syntax for the instructions is similar to a relay ladder logic diagram. Ladder allows to track the power flow between power rails as it passes through various contacts, complex elements and output coils (Figure 14).

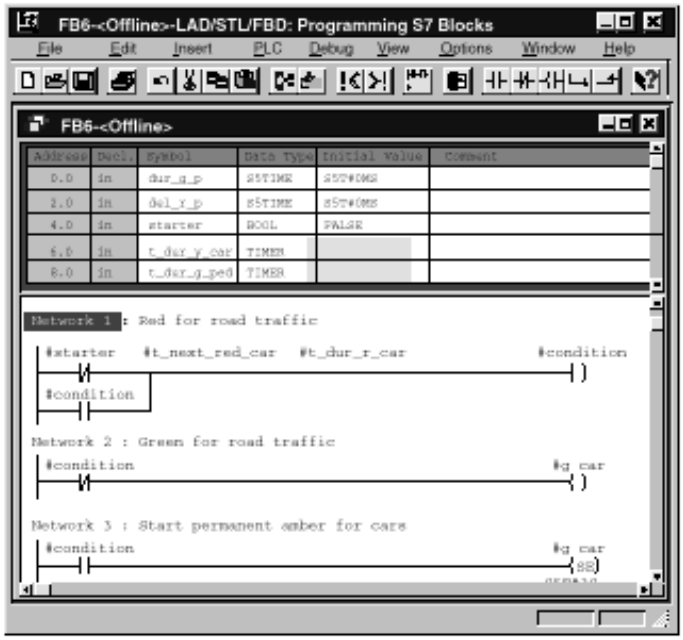

Figure 14: LAD programming.

- STatement List (STL) is similar to machine code. The instructions of a STL program correspond to the steps of the program executed by the CPU (Figure 15).

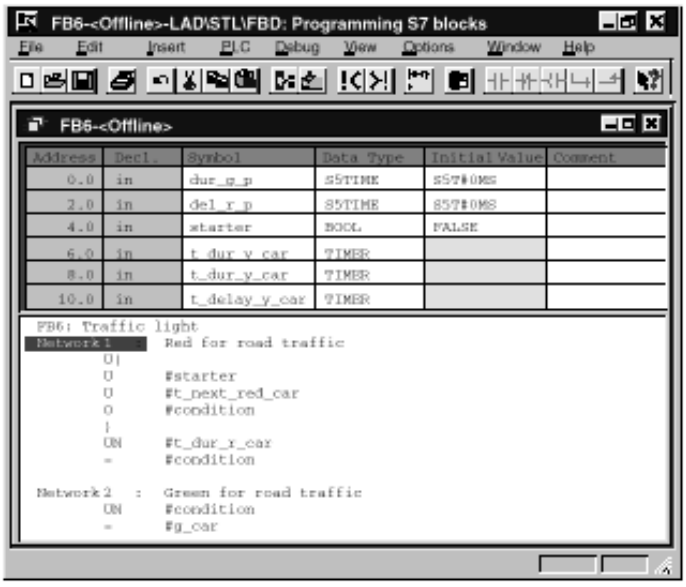

Figure 15: STL programming.

- Function Block Diagram (FBD). The syntax uses logic boxes, like AND, OR from boolean algebra to represent the logic (Figure 16).

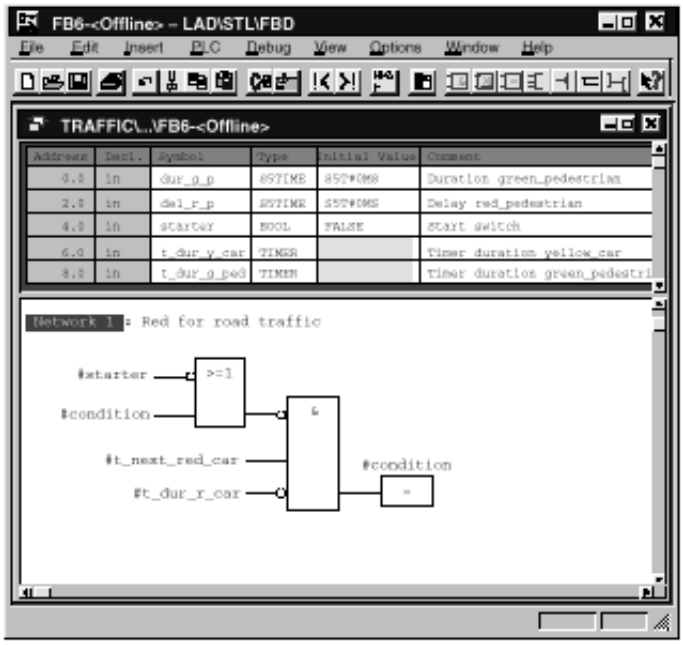

Figure 16: FBD programming.

Other programming languages are available as optional packages:

- S7 Structured Control Language (SCL). It contains language constructs similar to those found in the PASCAL and C programming languages. It can be used in combination with STL, LAD and FBD.
- S7 GRAPH. This is used to program sequential controls. The process sequence is divided into steps which contain action to control the outputs. The transition from one step to another one is controlled by switching conditions (Figure 17). An OB or FB written in STL, FBD or LAB calls the sequencer.

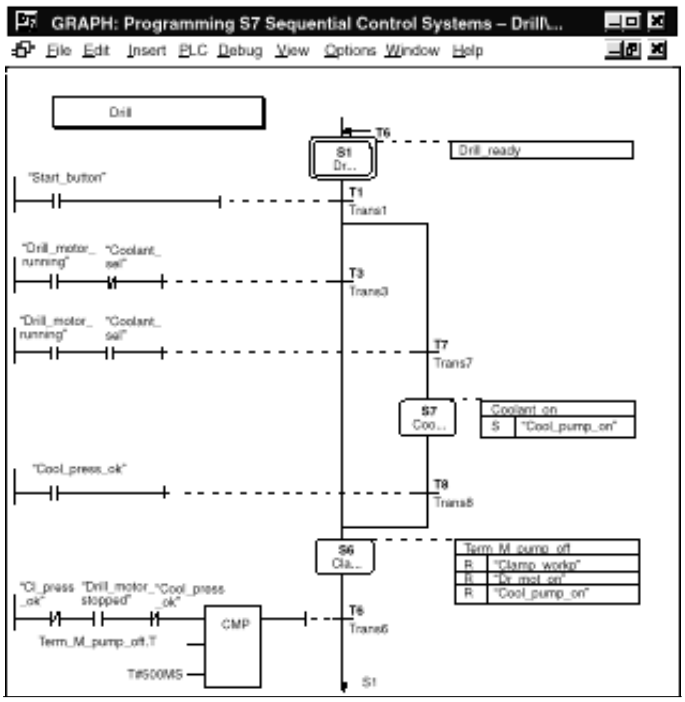

Figure 17: S7 GRAPH example.

- S7 HiGraph. This is used to describe asynchronous, non-sequential processes in the form of state graphs (Figure 18). Messages can be exchanged between different graphs to do synchronization. The graph has to be called by a cyclic OB. The graph can call FCs, FBs written in STL, LAD, FBD and SCL.

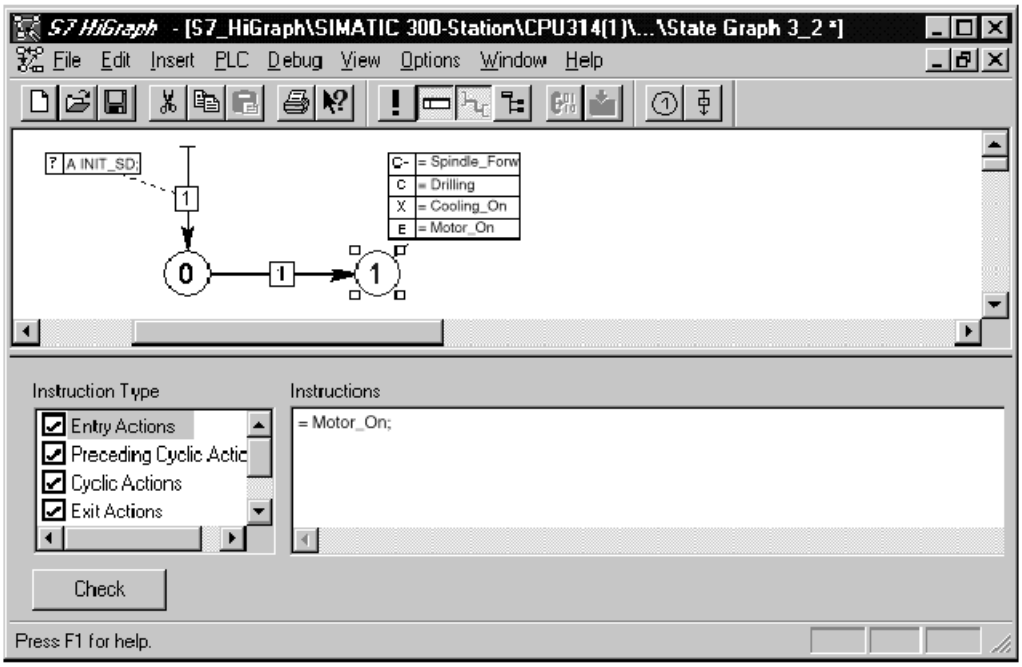

Figure 18: S7 HiGraph

- Continuous Function Chart (CFC): it is used to create an overall software structure for a CPU from predefined blocks (Figure 19). To do this, blocks are placed on function charts, configured and interconnected. Interconnecting means that values are transferred from one output to one or more inputs for the communication between blocks or common operands.

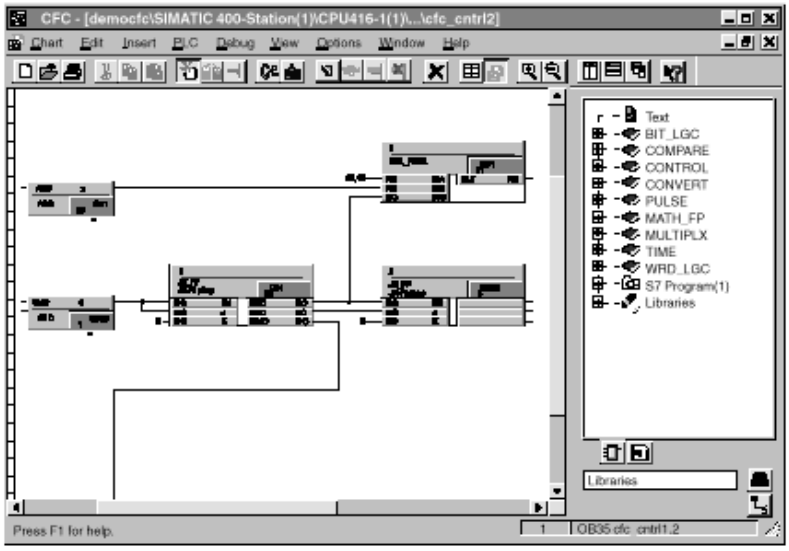

Figure 19: Graphic link with CFC

Library functions like the SEND/RECEIVE interface, or system functions like JUMP, MOVE, etc. are included in STEP 7 and accessible from all the languages.

Projects in STEP 7 represent the sum of all the data and programs within the scope of an automation task. They are used to store the data and programs in an organized manner. The data collected in a project include:

- configuration data for the hardware structure and the parameters of the modules where you drag and drop the desired hardware module (Figure 20)
- configuration data for communication on the networks (Figure 22): Ethernet, PROFIBUS, MPI
- programs for the PLC (Figure 21).

STEP 7 supports two modes: off-line and on-line from which one can use the same tools. In off-line one can download the software to the PLC, and in on-line mode, one can upload the running software from the PLC.

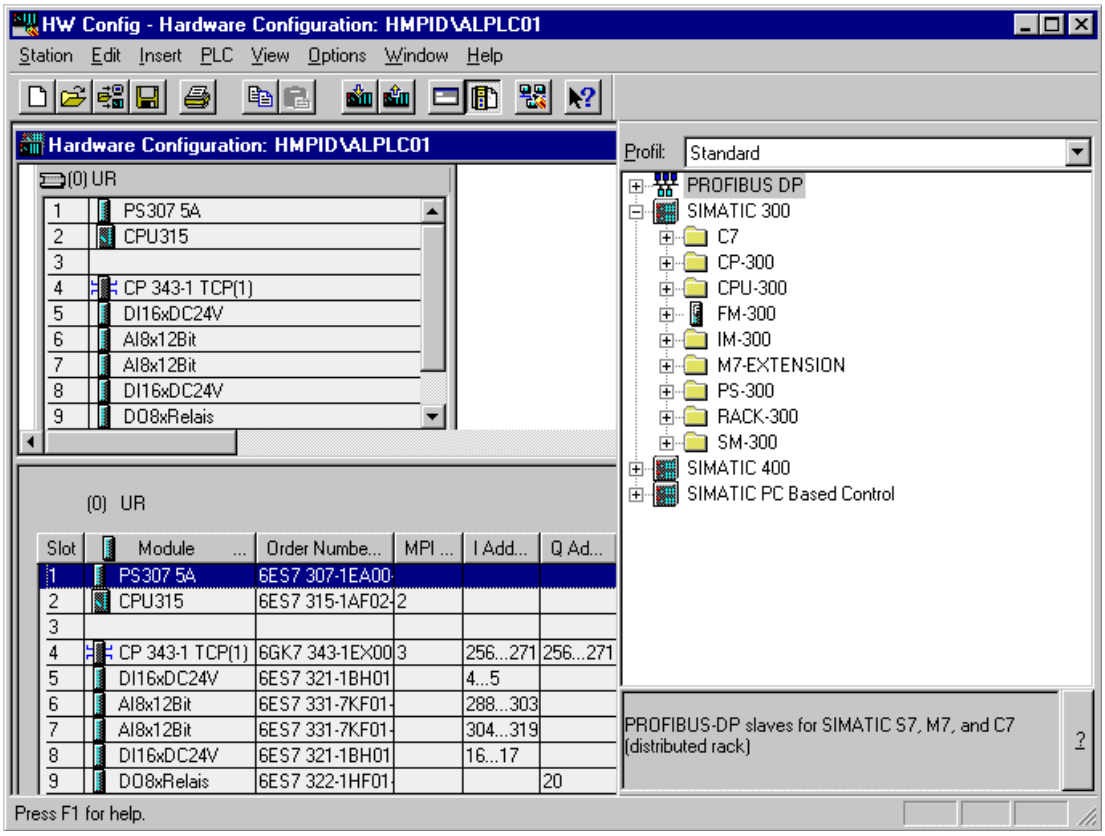

Figure 20: The STEP 7 tool to configure the hardware.

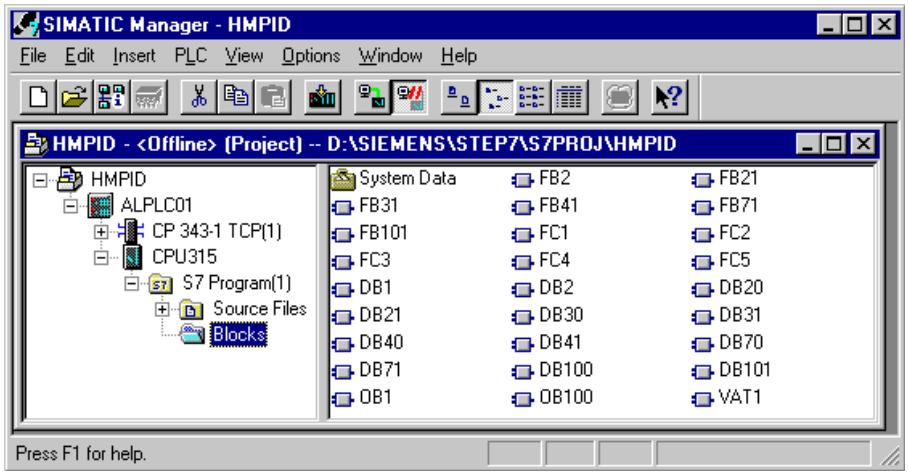

Figure 21: Blocks and functions of the HMPID PLC user program.

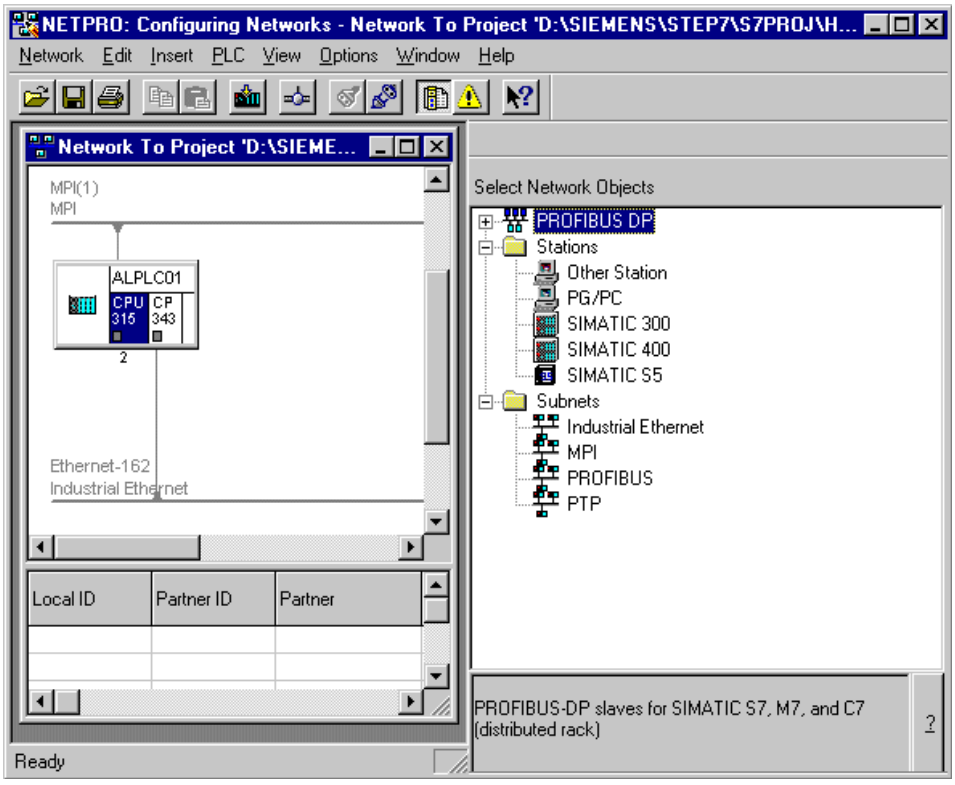

Figure 22: The STEP 7 tool to configure the networks.

The user program and the PLC configuration can be downloaded with STEP 7 by MPI, PROFIBUS or Ethernet. In the case of Ethernet, the IP number of the card has to be entered first by MPI bus.

A Graphical debugger is available in STEP 7. The variables: inputs, outputs, bit memory, counters, peripheral outputs and elements of DB can be displayed (monitored) and

modified. In STL language, the program is debugged via step by step execution and setting breakpoint or by displaying the content of the registers of the statements. FBD and LAD programs are debugged by showing the signal flow within the network of a block.

The CPU and I/O module diagnostic buffer in which are stored the online warnings or errors during execution is accessible within STEP 7.

3.3 OPC

OPC is an acronym for OLE for Process Control and is used as a standard interface for communication in automation engineering. OLE includes the component model of Microsoft. The OPC interface is the specification of a uniform and vendor independent software interface based on OLE. It was developed as an industrial standard by leading firms in the field of automation with the support of Microsoft.

Although OPC is primarily designed for accessing data from a networked server, OPC interfaces can be used in many places within an application. At the lowest level they can get raw data from physical devices into a SCADA or DCS, or from the SCADA or DCS system into other client application such as Business management, etc. An OPC server can be built to allow a client application to access data from different OPC servers from different OPC vendors running on different nodes (Figure 23), i.e. one to many relationship. Development kits for writing OPC Servers and OPC Clients are available for several platforms.

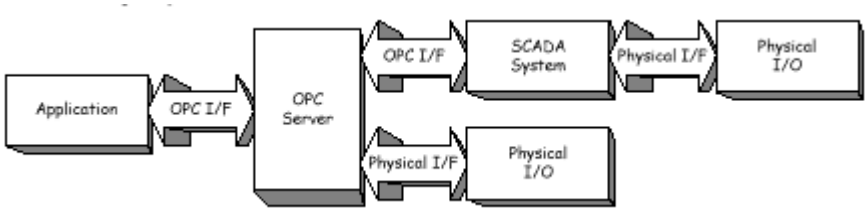

Figure 23: OPC client/server relationship

OLE is based on COM[12] (Component Object Model) from Microsoft. COM defines a standard that allows objects to be defined as separate units under Windows and to access these units beyond the limits of a process. Objects can be seen as extensions to the operating system, they are not dependent on programming languages and are available in principle to all applications. The COM component model has a client/server architecture. The server provides defined services to a client application that requests and uses the services.

With Windows NT, version 4, the COM specification was extended to DCOM[12] (Distributed COM) in order to allow objects also to be accessed outside the computer on which they are located. Objects used by an application can be distributed throughout a network.

OLE objects provide their services through defined interfaces (Figure 24). The implementation of an object, data and code, remains hidden to the user of the object. Each OLE object provides an “IUnknown” interface. All other interfaces in OLE are derived from IUnknown. The IUnknown interface provides clients with pointers to other interfaces for a given object and manages the behavior of an object.

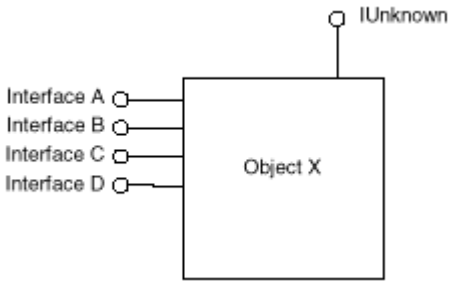

Figure 24: OLE object interface.

3.3.1 Interface types

An OPC server supports two different types of interface (Figure 25) used by the client to communicate with the server:

- Custom interface mainly used by C/C++ applications for maximum performance
- Automation interface used by applications based on scripting languages such as VB (Visual Basic).

OPC servers must implement the custom interface and optionally the automation interface.

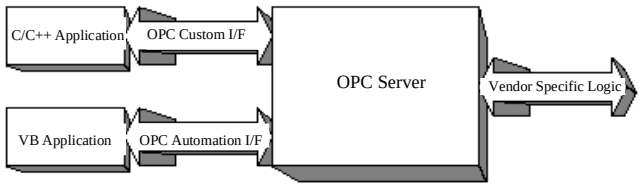

Figure 25: OPC interfaces

The OPC specification[8] defines the characteristics of a COM interface but not the implementation of such an interface. It specifies the behavior that the interfaces are expected to provide to the client applications. The specification expects that the server will consolidate and optimize data access requested by the various clients to promote efficient communication with the physical device. For input (Read), data returned by the device will be buffered for asynchronous dispatching or synchronous collection by different OPC clients. For output (Write), the OPC server updates the data send by the client on the physical device.

An OPC server includes three types of objects: the OPC Server object itself, the OPC Group object and the OPC Item object (Figure 26).

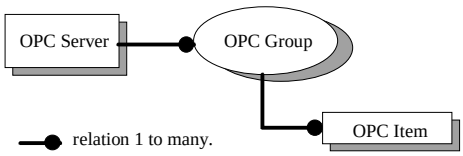

Figure 26: OPC object relationship.

3.3.2 OPC Item

The OPC Item object represents the connections to data sources within the server. An OPC item is not directly accessible as an object by an OPC Client. Therefore, there is no external interface defined for an OPC Item object. All access to an OPC Item passes via the OPC Group object that contains the OPC Item. Associated with each item is a Value, Quality (quality of the value) and Time Stamp attribute. The value is of type VARIANT (OLE terminology): real (VT_R4), signed char (VT_I1), unsigned char (VT_UI1), etc.

3.3.3 OPC Group

An OPC Group object includes the group object information and OPC items. Figure 27 is a summary of the OPC Objects and their interfaces. Some of the interfaces are optional indicated by [ ].

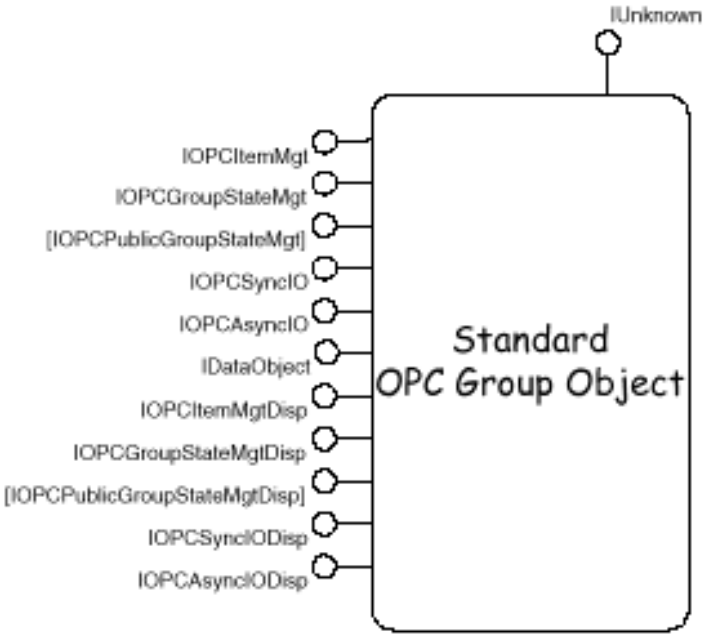

Figure 27: OPC Group object

The OPC Group has many attributes. Three of them are of particular interest:

- ActiveState

  An OPC Server can not only read and write the values of process variables but also monitor specific conditions. Monitoring means that the OPC Server checks whether the value of the process variable has crossed a deadband (see PercentDeadBand) after a time interval (see UpdateRate) has elapsed. In which case the value is automatically sent to the OPC Clients. Each OPC Group and each OPC Item have an ActiveState attribute. An OPC Client can disable or enable the ActiveState of an OPC item or an OPC Group.

- UpdateRate

  The UpdateRate is the interval at which the OPC Server checks the value of an OPC Item that is monitored.

- PercentDeadBand

  This deadband applies only to an OPC Item of type analog. The monitored value is only sent to the client if it has changed by more than the specified percentage of range (minimum and maximum of the value).

3.3.4 OPC Server

The OPC Server object holds information about server status, version, etc. It contains the OPC Group objects and provides the functionality to an OPC client to create and manipulate OPC Group objects. Figure 28 is a summary of the OPC Objects and their interfaces. Some of the interfaces are optional indicated by [ ].

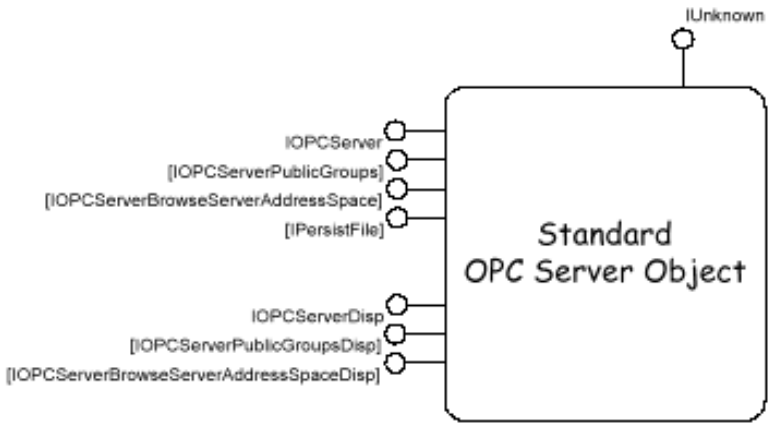

Figure 28: OPC Server object

CACHE data and DEVICE data are defined as part of the interface of the OPC Group object. Details for the implementation are, however, not given in the specification but it is assumed that most OPC Servers will implement some sort of CACHE. The CACHE should reflect the latest value of the data as well as the quality attribute and time stamp. Access to the CACHE data is expected to be fast against access to the DEVICE data that implies a complete acquisition cycle.

Each client application is responsible for storing persistently the relevant client configuration information (OPC Group and OPC item definitions). It is important to distinguish the address space of the server or server configuration like the OPC Item list and subsets of this space that a particular client may be interested in at a particular time (OPC Group and OPC Item). The implementation of the server address space will depend on the application and could be:

- entirely fixed (e.g. dedicated interface to a particular device)
- configured entirely outside of the OPC environment (e.g. interface to an existing external system)
- automatically configured at start-up by a server which can poll the existing system to inquire about installed hardware or interfaces
- automatically configured on the fly by a server based on the names of the data items the client applications are currently requesting.

It is assumed that the server address space is managed within the server.

3.3.5 OPC Client

An OPC Client directs the server to create, manage and delete the OPC Groups. The OPC Client can define one or more OPC Items in each OPC Group in order to access the desired data and to subscribe to a list of items. Public groups are treated like server defined groups or used for persistent storage by an OPC server. The concept allows to share configuration data information across multiple client applications. Public groups can be created where only one client application defines the OPC Items and other client applications access the information by connecting to the public groups.

Synchronization is the ability of a client to read or write multiple values and attributes in a single transaction. Serialization is the ability of a client to control the order in which write transactions are executed.

3.3.6 Access Security

OPC access security is based on the COM/DCOM scheme. There is no security specific coding inside the DCOM client/server application. Developers and administrators configure the security setting for each component using the NT configuration tool dcomcnfg. The three main parameters are:

- configuration: normally left to the administrator of Windows NT
- launch: defines which user or group of users is allowed to launch the server and retrieve objects
- access: defines which user or group of users is allowed to issue calls on the objects when the connection between a client and a server has been established.

No permissions can be defined for OPC Groups or OPC Items for different users within an OPC Server. If the OPC Server computer and the OPC Client computer are not in the same network domain (like the CERN domain for NICE) then users defined in the access and launch permission should be declared in the OPC Client computer. DCOM transmits the username of the calling client application to the remote computer where the corresponding component is running.

DCOM maintains a background ping between a client and a server. If a client connection is lost Windows NT will detect this within 6 minutes and all the interfaces the client had requested from the server will be released.

3.3.7 Integration of OPC in BridgeVIEW

BridgeVIEW can communicate with any server implementing the OPC server interface. It detects automatically all OPC Servers installed on the own PC and searches the network for OPC Servers on other computers. Unlike Device Servers, OPC Servers do not store information in the Common Configuration Database but the client needs to read all available information about server capabilities and items from the server directly.

Each BridgeVIEW I/O group (Figure 5) created in the Tag Configuration Editor is automatically mapped to an OPC Group in the OPC Server with the same attributes. The Item name of the BridgeVIEW tag (Figure 4) is the OPC ItemID (Item IDentificator).

BridgeVIEW uses intensively the Active mechanism (See ActiveState) of OPC. The OPC Groups and OPC Items are saved in the .scf file. It is not possible with the BridgeVIEW OPC Client to request a “Read” on OPC Items.

3.3.8 SIEMENS OPC Server

SIEMENS provides an OPC Server for SIMATIC NET on Windows NT. This server allows simple access to SIMATIC NET networks for any Windows application that supports the custom or automation OPC interface. The SIEMENS OPC Server connects to the PLC via PROFIBUS or Ethernet using the S7 database configured by the COML S7 tool (Section 3.2.2). The ItemID is the identificator of the OPC Item. It is a string with the following structure: [Protocol-ID:<connection-name>]variable-name. The Protocol-ID is DP for PROFIBUS and S7 for S7 communication over PROFIBUS or Ethernet. The management of the connection is handled by the OPC Server and is hidden from the user. The connection name consists of three parts separated by the character “|”:

- the specified name of the S7 connection (e.g.: AL-PLC) defined in the S7 database
- the VFD at which the OPC Server will log on (e.g.: VFD) defined in the S7 database
- the name of the communication processor (e.g.: S7ONLINE when using the SOFTNET package for connection via an Ethernet PC card)

The OPC Server allows access to the following data: data blocks (DB), inputs (I), outputs (Q), peripheral inputs (PI), peripheral output (PQ), memory bit (M), timers (T) and counters (C). Three other items are defined for handling the communication and connections:

- &identify(): provides information about the attributes of a device
- &status(): provides the status of the device
- &statepath(connection-name): provides information on the status of the connection with the remote device.

OPC Items which are not in the address space of the OPC Server (OPC Item list) are dynamically created when an OPC Client connects to them.

The OPC Server for SIMATIC NET can support simultaneous operation on the DP and S7 protocol. The S7 OPC Server can use two different procedures of the S7 protocol to obtain the value of a process variable. It can poll the values of the variables at a selected interval

by sending read requests to the remote partner devices or use the “Remote from Partner Device” mode. The partner reports the values of the required variables automatically to the OPC Server at selectable intervals. This reduces the load on the OPC Server. However, the number of variables that a remote partner can report is limited by the resources of the PLC.

4 Comments on Hardware and Development tools

The hardware and software environment was entirely new to the team. The following comments reflect, therefore, also the experience gained during the learning phase.

4.1 PLC and input/output

PLCs are well adapted to a two level architecture where the control process must be independent of the supervisory system and where the system (I/O readout, closed loop control, etc.) needs to be reliable and independent from the communication network or a remote computer.

The installation of the PLC and the input/output modules is straight forward and fast. Accessing the IOs does not require any programming. The analog input modules are configurable for a wide range of parameters like voltage, current, resistor, pt100, etc. The connectors are state of the art technology and are specific for each module type. The values at the analog inputs are continuously read and converted to digital. Different filters for noise rejection can be selected with the configuration tools.

The number of input/output modules connected to the CPU bus is limited and depends on CPU type. A maximum of 8 modules can be connected in the same chassis and up to 32 with a rack extension. A wide range of analog readout modules exist. But SIEMENS digital input/output modules with a CPU bus connection are restricted to 0-24V or 220V/130V. No TTL input/output modules are available. Without the optional battery backup the user program has to be downloaded after every power off.

Detailed knowledge of the PLC architecture and functionality is required to develop a user program. Initially the people involved had no knowledge of SIEMENS PLC S7. The documentation on the PLC and a “quick start” manual are a good help to shorten the learning time. Only half a day was necessary to configure correctly the input/output modules and to read/write digital and analog values. But the documentation on writing a user program, calling FCs or FBs or to use the return value of the calling block, etc. is incomplete and unsatisfactory.

The SIEMENS PLC operates in polling mode with defined execution cycles. The OBs and the number of blocks (FBs, FCs and DBs) which can be created and the programming languages depend on the CPU type but all S7-300 CPUs support STL, LAD and FBD. To read analog and digital inputs at different scan rates requires complex programming. The digital inputs are read before every CPU cycle, and the digital outputs are written after every cycle. The analog inputs are not automatically read by the CPU, the user program must access the PI with the corresponding address to get the analog values and PQ to write analog values. Data read through Ethernet or read from PROFIBUS have to be transferred into the CPU memory through SFC or SFB blocks. The Ethernet connection seems stable but no tests were made for PLC to PLC communications.

4.2 STEP 7, languages and tools

STEP 7 is the SIEMENS development software to configure and program the S7 PLCs. It is an intuitive graphical tool. The FBD, STL and LAD languages are include in the basic version. Other languages are optional and must be purchased. They are not recommended for small S7-300 CPUs. STEP 7 allows to archive and retrieve projects and to create shared libraries. The tool is powerful and easy to use. There is also a utility to connect to the PLC and debug the hardware.

Starting to work with STEP 7 is easy and fast. Two weeks only were needed to communicate with the PLC and understand how it works, use STEP 7 and its tools, write a small program in the FBD language, run it, debug it and test it. A one day initiation from a person with some experience of STEP 7 helped, however, significantly.

The library provided with the languages are quite extensive. However, documentation on functions and on the CPU registers is incomplete. Dynamic checking of the connection type is performed during the writing of user programs with FBD. Symbolic addressing of variables used in any block inside a project can be used instead of direct memory access. Step by step debugging is only available for the STL language. A FBD program can be debugged by analyzing the signal flow inside the network of a block. Unfortunately, the debugger shows just the signal flow of the block which is visible in the open window. A “step into” facility when calling a block from another block is missing. However, the run-time diagnostic buffer of the CPU and the I/O modules can be accessed for debugging.

The S7 database in the PC for PC to PLC communication is created with the COML S7 tool. The relevant documentation has, unfortunately, be found to be not very clear.

4.3 OPC

The SIEMENS OPC Server polls the PLC. Fast polling rate and PC CPU load have to be balanced for best system use. The OPC Items are dynamically created in the Server when an OPC Client requests an “AddItem” for an Item which does not exist. Unfortunately, they cannot be deleted. One solution is to re-start the OPC Server with a new S7 database.

In the HMPID prototype, the OPC Server and the OPC Client run in the same PC. Successful tests were made in the laboratory to access the OPC Server from another computer.

The OPC Server seems to be reliable. It is able to detect and recover from a connection failure with the PLC. No tests on performance, reliability, etc. were done in the context of this project but information is available from the OPC foundation WEB site[8].

4.4 BridgeVIEW

The fundamentals of BridgeVIEW are rather easy to learn, but for more advanced features more detailed documentation would be desirable. The development of a HMI (Human Machine Interface) is straight forward. A majority of the supervisory facilities are readily implemented, but the collection of supplied front panel objects for animation is rather restricted. Adjusting the size of objects in the front panel is very tedious especially for small objects, e.g. when the size of an object needs to be related to the size of some other object. Moreover, resizing the panel window does not resize the objects as well.

The configuration tools (tag configuration editor, server explorer) are user friendly. Exporting and importing configurations is simple and was used to run a copy of the application on a remote OPC Server.

During the development, a VI Test Server developed for the TRT Gas prototype was used to simulate the SIEMENS OPC Server. The BridgeVIEW program could therefore be developed faster without direct link to the PLC.

Wizards, inside BridgeVIEW, allow to create the G programs for the HMI. They are very helpful to connect front panel controls or indicators with tags and to create diagrams to open and close panels. A control, in BridgeVIEW terms, is an object that can receive a value and can be connected to an input/output tag. The underlying G code periodically checks if the value of a control is different from the last value sent to the BridgeVIEW server (OPC Client) of the tag. In this case the new value is sent to the server and written into the RTDB when the BridgeVIEW server reads it. If the BridgeVIEW server is unable to send this value to the hardware, the same value can only be sent again after modifying the wizard code or by setting a different value first and then subsequently the initial one.

The BridgeVIEW panel, i.e. the graphical interface including the control object is executed periodically even if the value has not changed which is very inefficient since BridgeVIEW needs to evaluate each time the value of the control and to execute the G code connected to this control.

BridgeVIEW provides also a wizard for creating the G code for the alarm panel. However, the generated G code had to be edited to include also the acknowledgment of the alarms.

Similarly, tags that are not already initialized when the engine is started do not generate alarms. Therefore, the code generated by the wizard was modified in order to show uninitialized values to the operator.

4.4.1 BridgeVIEW and OPC

The OPC Client of BridgeVIEW is using the Active mechanism of OPC (See ActiveState). It is not possible to read periodically an OPC Item. In the “OPC DCOM White Paper”[8] available on the OPC foundation web site, it is written that an OPC Client can detect a failure of the OPC Server by periodical calls like ‘IOPCServer->GetStatus()’. BridgeVIEW does not implement that protocol and therefore it is unable to detect OPC Server failure. An alarm error status on the tag is only raised, as soon as the client tries to write to an OPC Item otherwise no alarms are generated. In addition, BridgeVIEW can only connect again to the OPC Server after the engine is re-started.

5 Conclusion

PLCs are a suitable solution where the control process must be independent of the supervisory system and where the system (I/O readout, closed loop control, etc.) needs to be reliable and independent from the communication network or a remote computer. The programming tools are intuitive and easy to use.

BridgeVIEW is adequate for the development of supervisory applications for small prototypes. It is flexible enough to develop the user interface required for the prototype HMPID liquid distribution.

OPC looks promising as a common interface between systems, especially as it allows at a later stage to migrate smoothly to another supervisory system. Whereas some documentation exists on the OPC foundation web page[8], more investigation will be necessary to study reliability, performance and general capabilities.

The ALICE-HMPID liquid distribution prototype application has a modular structure that will allow expansion to the full HMPID detector. The software at the level of the control functions in the PLC and also the supervisory level has been designed to be adaptable to different control strategies and is scalable to larger applications. The system has been tested to the full satisfaction of the users from the point of view of control functionality and the user interface. Improvements and extensions to the functionality are foreseen and will be implemented as result of operational experience.

Although the application developed was provisionally limited to a detector prototype, the experience gained will be valuable for the control of the LHC experiments.

Acronyms

OPC.	OLE for Process Control
SCADA.	Supervisory Control And Data Acquisition
DCS.	Distributed Control System
PLC.	Programmable Logic Controller
HMPID.	High-Momentum Particle IDentification
RICH.	Ring Imaging CHerenkov

References

1 ALICE URL:
http://www.cern.ch/ALICE/

2 ALICE DAQ, DCS
http://www.cern.ch/ALICE/Projects/Detector_Control_System/
http://aldwww.cern.ch/

3 Test and Evaluation Station (TESt), A control system for the ALICE-HMPID liquid distribution prototype
ALICE/99-13 Internal note-DCS

4 JCOP home page URL:
http://itcowww.cern.ch/jcop/

5 ALICE HMPID TDR URL:
http://www.cern.ch/ALICE/TDR/HMPID/

6 National Instrument home page URL:
http://www.natinst.com/

7 SIEMENS:
http://www.siemens.de/en/

8 OPC foundation:
http://www.opcfoundation.org/
http://www.opcfoundation.org/opc_public_tech.htm

9 Fieldbuses recommendation

http://itcowww.cern.ch/fieldbus/report1.html
10 PLC recommendation
private communication
11 PROFIBUS:
http://www.profibus.com/
12 OLE-COM-DCOM:
http://www.microsoft.com/activex/default.asp

Acknowledgment

We would like to thank F. Michaud (IT/CO), P. Baehler (I/CO) who helped us with BridgeVIEW, R. Barillere (IT/CO) for his useful comments on BridgeVIEW and J.R. Sendon del Rio (IT/CO) for the help and advice on OPC.