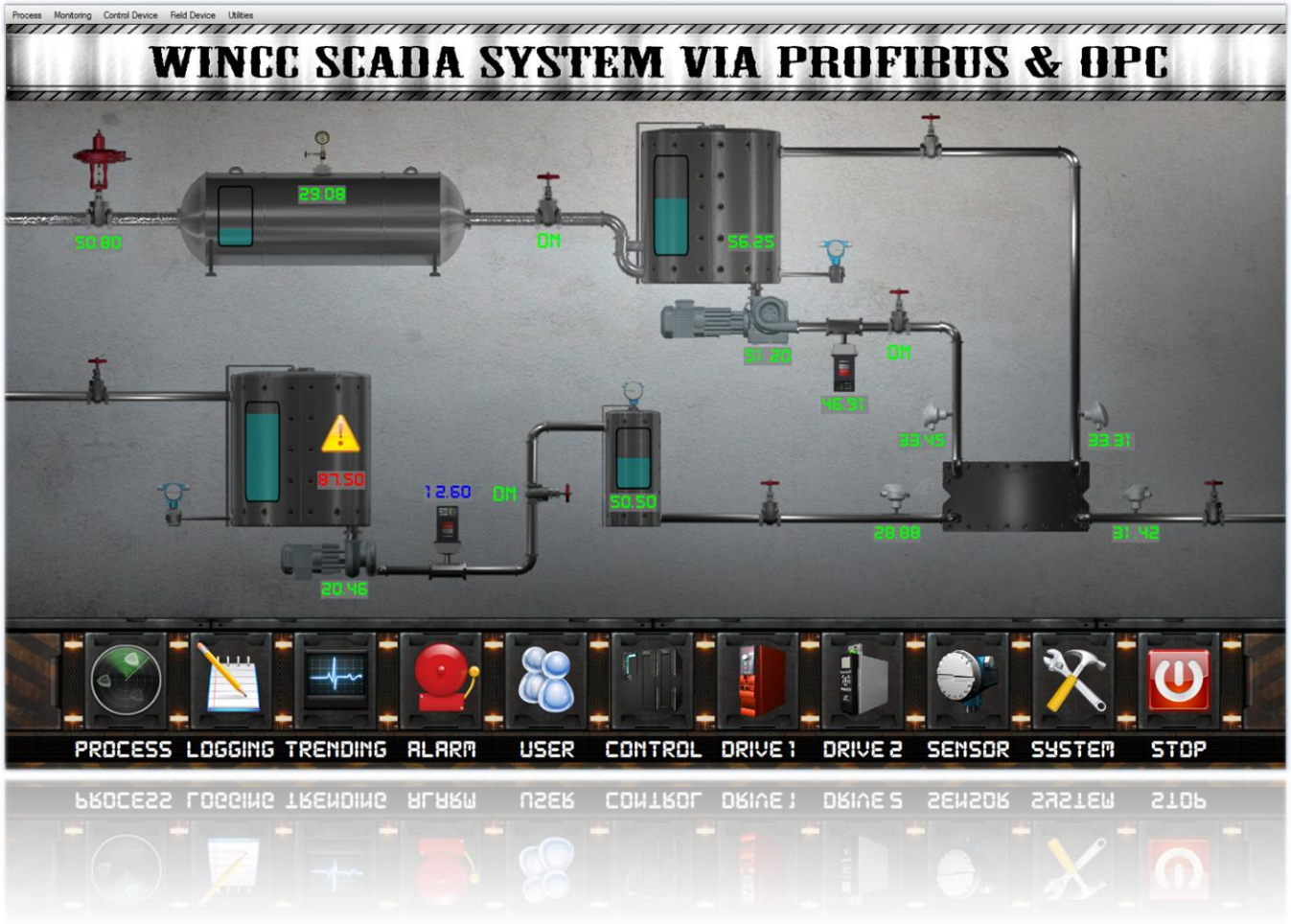
WINCC SCADA SYSTEM VIA PROFIBUS & OPC



"A report submitted to the School of Engineering, Murdoch University in partial fulfilment of the requirements for the degree of Bachelor of Engineering"

Project Developer: Hao Xu
Project Supervisor: Graeme Cole

MURDOCH
UNIVERSITY

This page is internationally blank.

"When heaven appoints a person to undertake a great mission, he must first of all suffer from a spiritual pain, undergo physical hardships and be distracted from his action. In this way, his soul will be touched upon and he can develop a spirit of perseverance and endurance. Then he will be stengthened to perform what he could not perform ever before !"

------- *Mencius*

ABSTRACT

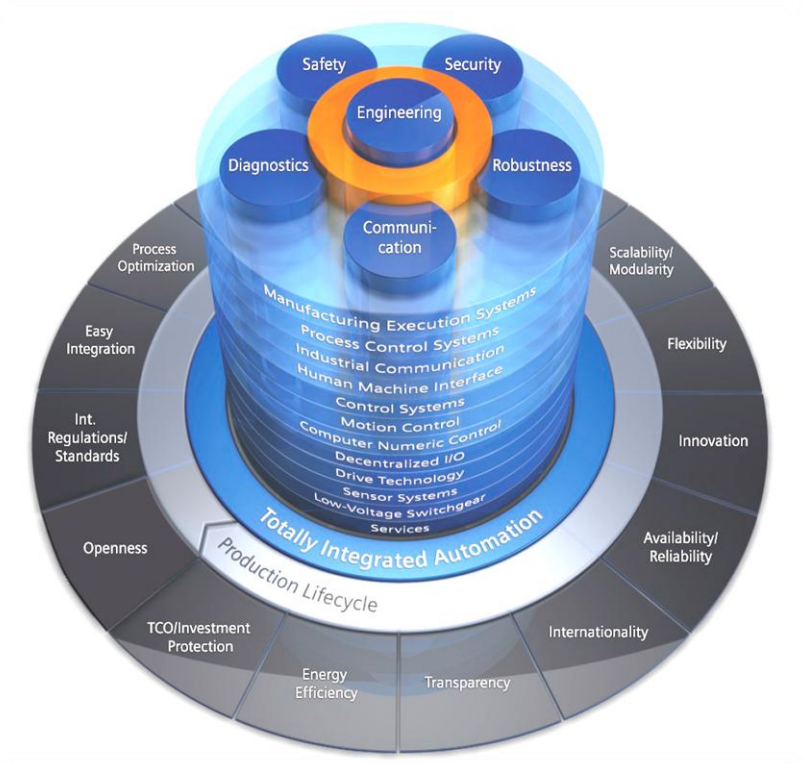


Figure 1: Totally integrated automation concept [38]

Over 1500 hours' effort were put into this intensive project with 6 weeks prior to the commencing date of the project to meet the objectives and requirements of the project. During this project, all the available devices, instruments including their configuration software were properly configured with all the expected features fully functioning. This thesis report summaries most of the work done with a reasonably detailed level to support future students with a comprehensive background to develop the project aspects in the future.

A comprehensive SCADA system was designed and implemented using the configured devices and the control modules in the laboratory. The network structure of the control system was relatively large, but indeed demonstrated the capability of the automation system.

Profibus communication networks in the laboratory were carefully designed and fully set up with proper labels, which would then be used as an education tool to gain the experience of industrial communication by the students.

Over 500 pages of configuration manuals of devices and configuration software were created as a large part of the focus of this project. The configuration manuals contain a huge range of information which covers more than what students need during their study. Those manuals are expected to guide students for their projects and troubleshooting.

ACKNOWLEDGEMENTS

The acknowledgements are given to the following individuals for the assistance during the project:

Project supervisor: Graeme Cole

Technical officer: William Stirling

Technical officer: John Boulton

Technical officer: Lafeta Laava

ACRONYMS

A&E	Alarms & Events	**MAV**	Main Actual Value
AC	Alternate Current	**MBP**	Manchester Bus Powered
AEO	Automatic Energy Optimization	**MCT**	Motion Control Tool
AI	Analog Input	**MMC**	Micro Memory Card
AMA	Automatic Motor Adaptation	**MPI**	Multi Point Interface
ANSI	American National Standards Institute	**MRV**	Main Reference Value
ASCII	American Standard Code for Information Interchange	**MS**	Master Slave
AO	Analog Output	**MS**	Module Status
API	Application Programming Interface	**MSB**	Most Significant Bit
AS	Automation System	**NC**	Normally Closed
AS	Autosetup	**NEMA**	National Electrical Manufacturers Association
AS-I	Asynchronous Serial Interface	**NO**	Normally Open
BCC	Block Check Character	**NS**	Net Status
BCD	Binary Coded Decimal	**OBW**	Override Bandwidth
BF	Bus fault	**OLE**	Object Linking and Embedding
CAN	Controller Area Open	**OPC**	OLE for Process Control
CBA	Component Based Automation	**OS**	Offset
CCW	Counter Clockwise	**OS**	Operating System
COM	Component Object Model	**P & ID**	Pipe & Instrumentation Diagram
CP	Communication processor	**PA**	Process Automation
CPU	Central Processing Unit	**PB**	Profibus
CS	Configuration System	**PC**	Personal Computer
CS	Control Select	**PCA**	Parameter Characteristics
CSV	Comma-Separated Values	**PCD**	Process Control Data
CTW	Control Word	**PCI**	Peripheral Component Interconnect
CW	Clockwise	**PCV**	Parameter Characteristics Value
DA	Data Access	**PD**	Process Data
DB	Data Block	**PDU**	Protocol Data Unit
DC	Direct current	**PI**	Process Input
DCOM	Distributed Component Object Model	**PID**	Proportional-Integral-Derivative
DIP	Dual In-line Package	**PLC**	Programmable Logic Controller
DLL	Dynamic-Link Library	**PN**	Profinet
DMC	Dynamic Matrix Control	**PNO**	Profibus Nutzerorganization
DO	Digital Output	**PNU**	Parameter No.
DP	Decentralize Peripheral	**PO**	Process Output
DS	Data Set	**PPI**	Point-to-Point Interface
DSM	Distributed System Manager	**PPO**	Parameter Process data Object
DST	Daylight Saving Time	**Profibus**	Process Field Bus
EDD	Extended Diagnosis Data	**PVA**	Parameter Value
EEPROM	Electrically Erasable Programmable Read-Only Memory	**PWM**	Pulse Width Modulation
EMC	Electromagnetic Compatibility	**RAM**	Random Access Memory
EPROM	Erasable Programmable Read-Only Memory	**RC**	Request/Response Characteristic
ES	Engineering Station	**REF**	Reference
ETR	Electronic Thermal Relay	**RFI**	Radio Frequency Interface
FB	Function Block	**RJ**	Registered Jack
FBD	Function Block Diagram	**RS**	Recommended Standard
FC	Function Call	**RT**	Runtime
FMS	Field Message Specification	**RTD**	Resistance Temperature Detector
GPIO	General Purpose Input/Output	**RTU**	Remote Terminal Unit

GSC	Global Script Control	**SBus**	System Bus
GSD	General Station Description	**SBW**	Staging Bandwidth
GUI	Graphical User Interface	**SCADA**	Supervisory Control And Data Acquisition
GW	Gateway	**SCL**	Structured Control Language
HDA	Historical Data Access	**SD**	Start Delimiter
HSA	Highest Station Address	**SF**	System fault
HMI	Human Machine Interface	**SFB**	System Function Block
HTTP	Hypertext Transfer Protocol	**SFC**	Sequential Function Chart
HVAC	Heating, Ventilation and Air Conditioning	**SI**	System International
HW	Hardware	**SISO**	Single Input Single Output
HY	Hysteresis	**SLC**	Smart Logic Control
I/O	Input/Output	**SMP**	Spontaneous Message
ICE	Instrumentation & Control Engineering	**SQL**	Structured Query Language
ICSE	Industrial Computer System Engineering	**STL**	Statement List
ID	Identity	**STW**	Status Word
IEC	International Electrotechnical Commission	**SW**	Software
IGBT	Insulated Gate Bipolar Transistor	**TCP/IP**	Transmission Control Protocol/Internet Protocol
IIS	Internet Information Server	**TF**	Torque Flange
IL	Instruction List	**TIA**	Totally Integrated Automation
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench	**TOF**	Time Of Flight
LAD	Ladder Logic	**USB**	Universal Serial Bus
LAN	Local Area Network	**VBA**	Visual Basic for Application
LCD	Liquid-Crystal Display	**VBS**	Visual Basic Script
LCP	Local Control Panel	**VSD**	Variable Speed Drive
LD	Ladder Diagram	**WinCC**	Windows Control Centre
LED	Liquid-Emitted Diode	**WPF**	Windows Presentation Foundation
LSB	Least Significant Bit	**XML**	eXtensible Markup Language

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1: INTRODUCTION

The initial goal of this project was to design and implement a Siemens WinCC SCADA system using available Profibus interface devices. However, the scope of the project was expanded to integrate other systems or devices. This project has been developed by a few previous project developers and also been discovered by some group students in another unit. This project grew slowly due to the compatibility of the update software and newly purchased devices. The challenge for this project is that it involved a lot of work and required a huge amount of time and there is certainly no shortcut to achieve certain goals.

The device configurations done by the previous project developers were under the STEP 7 environment, but recently, the STEP 7 configuration software was updated to a new generation of software called TIA Portal. With TIA Portal, it was relatively easy to perform all the configurations, but it also meant all the previous work could not be properly used due to the totally different interface with workflow.

In addition, most of the devices have only been configured to have minimum capabilities from the previous project developers so the functions that the devices are able to perform is unknown before this stage. On the other hand, the Profibus networks in the laboratory were extremely confusing and there was no diagram from the previous documentation to illustrate the connection on the roof or behind the I/O panels.

Therefore, during this project, all the issues and uncertainties of the project were needed to be solved and settled so that students can have a fully working environment to work with later on.

CHAPTER 2: PROJECT OBJECTIVES

This project includes a lot of tasks and the contents that could be integrated into the project is enormous. However, the objectives can be summarized into 4 sections and are shown in **Figure 2**.

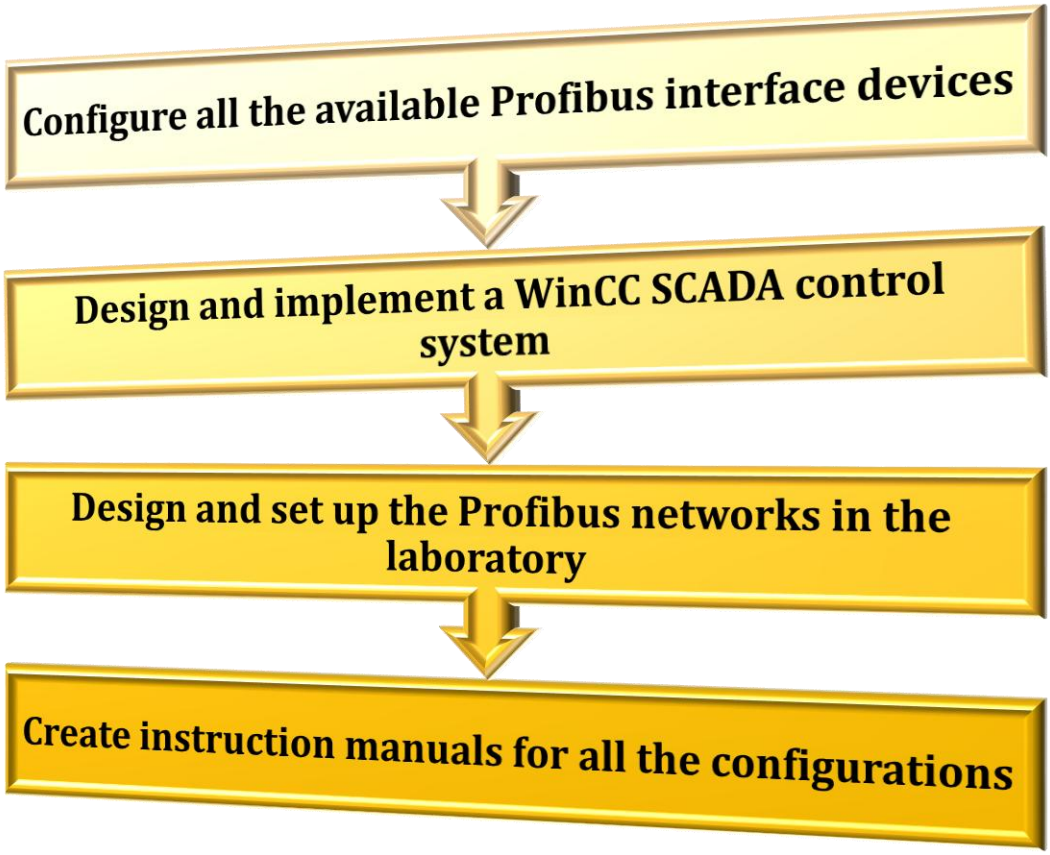


Figure 2: Outline of project objectives

Profibus Device Configuration

There are a range of Profibus interface devices and instruments available in the laboratory, but they have never been properly configured or integrated as part of the system. Most of those devices have been purchased for a few years and the functionality of them have not been discovered. So now it is urgent to find out whether they can be used as part of the system in the laboratory. Therefore, the configuration of the devices has the highest priority.

SCADA System Design and Implementation

Even though the WinCC SCADA system had been used by the previous project developer, the large range of the functionalities of the SCADA system had not been fully discovered. Hence, this was a good opportunity to get familiar with the industrial SCADA system and also explore the capability of the automation system.

Profibus Networks Setup

The Profibus networks in the laboratory were set up a couple of years ago and had never been tested and used for any exercise by students. The network structure and connections had not been documented which introduced some difficulties to be directly usde by the students. Hence, one of the objectives of this project is to test the connection of the Profibus PA network in the ICE (Instrumentation & Control Engineering) laboratory and redesign the Profibus DP network in the ICSE (Industrial Computer System Engineering) laboratory due to the increasing demand of the students.

Documentation

As the contents of this project are an extremely valuable opportunity for students to gain the peripheral device configuration experience and also learn how fieldbus and SCADA system work, the documentation is considered as an essential medium to pass the knowledge onto future students. This objective might be the most significant one because all the effort put into this project will not be wasted, but build a solid foundation for this project to grow.

CHAPTER 3: BACKGROUND

Data Communication

Data communication is the act of transmitting data between two communication partners. The data transferred serves different purposes depending on the applications. A SCADA system, in general, has the following communication options:

- Control and display status of the communication partners;
- Archiving;
- Reporting. [85]

Subnet

A subnet defines all the necessary physical components required to build a data transfer route or data exchange procedure. [85]

Connection

A connection is known as a correctly configured logical assignment of two communication partners for the purpose of a communication service. [85]

Topology

Topology describes the physical structure that devices or stations are connected to within the network. Different fieldbuses may support different topologies. [7] [85]

Protocol

The protocol can be described as an agreement between communication partners to implement communication services. A protocol typically specifies the structure of data communication on the physical link and also defines the operating mode, connection procedure, transmission rate, etc. [85]

Fieldbus

Fieldbus is a generic term used to describe the new digital industrial communication networks. The fieldbus is a bi-directional, multidrop serial communication network consists of controllers, actuators and transmitters. It is expected to replace the previous 4-20mA analog signal technology to satisfy the demand of complex control system.

Configuration

A configuration is an arrangement of functional units according to their major characteristics. In data communication, configuration pertains to the choice of hardware, software. The configuration highly affects the functionality and also determines the performance of the system.

ICE Laboratory

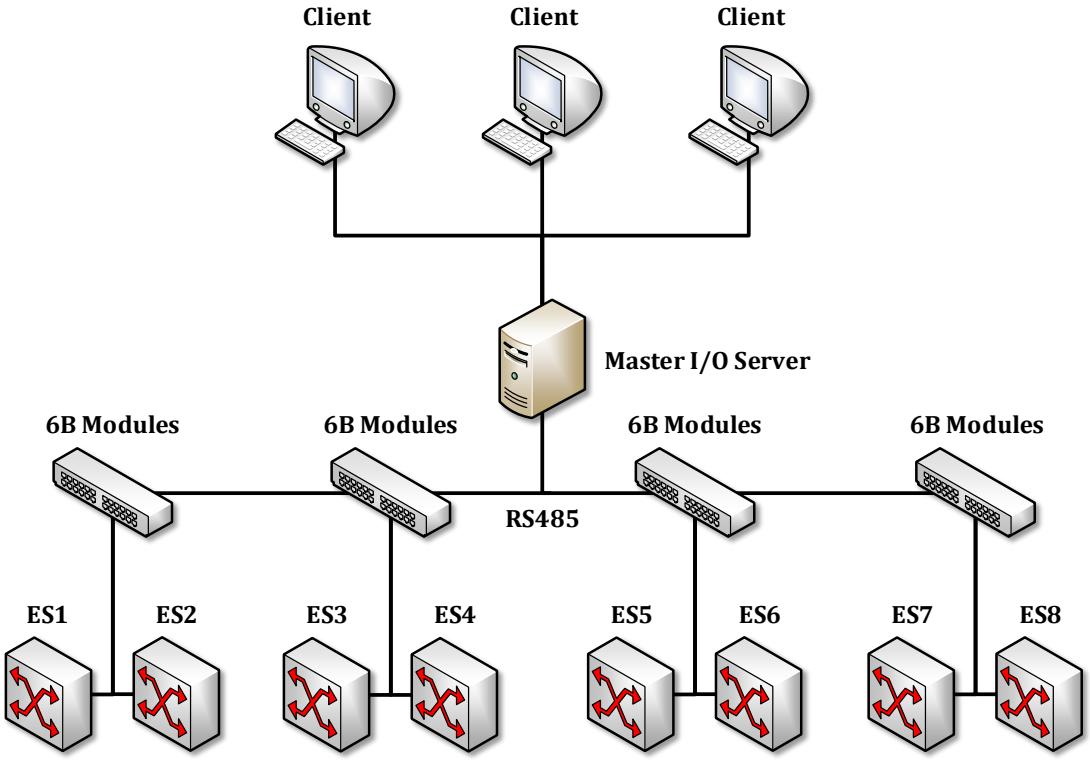


Figure 3: ICE laboratory master I/O server structure [18]

The Instrumentation and Control Engineering (ICE) laboratory in Murdoch University comprises a series of portable control modules and instruments such as control valves, pumps, air/water tanks for students majoring in ICE to gain some valuable experience. The essential control elements are control valves for air/water flow and are powered by the instrument air. A LabVIEW Master I/O server program is running on the server computer to collect the data from client computers. The data are then exchanged with 6B distributed I/O modules via RS485 communication. The 6B modules then send the signals to the control valves to implement automation. **Figure 3** shows the network structure of ICE laboratory. [18] [114]

ICSE Laboratory

The Industrial Computer System Engineering (ICSE) laboratory in Murdoch University is equipped with different kinds of controllers, actuators and measuring devices. Students enrolled in ICSE are required to use those devices to build various complex control systems. A Profibus DP network was set up earlier which connected all the panels in the laboratory.

CHAPTER 4: COMMUNICATION PROTOCOLS

MPI

The Multi Point Interface (MPI) is an interface between the CPU and a PG/OP or an interface for MPI subnet communication. MPI only supports transmission speed of 19.2kbps (only for S7-200 CPU) and 187.5kbps. [85]

Profi-Family

Profinet

Figure 4: Profinet logo [14]

Profinet (**Figure 4**) is an industrial Ethernet based communication standard for the automation industry. Profinet uses protocols such as TCP/IP and XML to implement communication, configuration and diagnosis for automation systems. Profinet comprises of 2 function classes which are Profinet IO and Profinet CBA (Component Based Automation). Profinet IO is used to exchange data and Profinet CBA is designed for distributed industrial automation applications. [14]

PROFIsafe

Figure 5: PROFIsafe logo [13]

PROFIsafe (**Figure 5**) is a safety communication technology used for fail-safe applications over Profibus and Profinet. PROFIsafe is designed to reduce the probability of data transmission error such as repetition, loss, insertion, delay and so on. PROFIsafe has 2 alternative implementations due to the transmission errors which are as follows:

- A bus line and a combined standard safety-oriented controller;
- Individual transmission lines and controllers for standard and safety-oriented automation. [13] [32]

PROFIdrive

Figure 6: PROFIdrive logo [10]

PROFIdrive (**Figure 6**) is a drive technology standard profile developed by drive manufacturers for Profibus and Profinet to minimize the commissioning time. PROFIdrive has defined 6 specific application classes which can be implemented independently to satisfy the wide range of industrial applications. PROFIdrive has 3 models which are base model, parameter model and application model. These 3 models are the core of the PROFIdrive to supervise the drive behaviour and assist the controller. [10] [32]

Profibus

Figure 7: Profibus logo [1]

Profibus (**Figure 7**) was created back in 1989 by the German government and several automation manufacturers. It was designed for high speed serial I/O in automation and industry applications. It is an open standard and was also known as the fastest fieldbus. In a Profibus network, a master controls and monitors all the distributed slaves in a multi-drop fashion over an RS485 serial bus. [1]

Profibus Variants

Profibus FMS

Profibus FMS stands for Field Message Specification. It is a complex universal protocol suitable for a wide range of applications at the cell level. It supports baud rates up to 500kBaud.

Profibus PA

Profibus PA (Process Automation) is essentially designed for applications within an hazardous explosive environment according to its intrinsic safety characteristic. It provides a connection between the central control system and the process automation system. All the PA field devices are bus powered and the fixed data transmission speed is 31.25kbits/s. [17] [85]

Profibus DP

Profibus DP (Decentralized Peripherals) is a fast and efficient communication protocol primarily designed for automation systems and distributed device communications. Profibus DP supports up to 244 bytes of input/output data per message with the highest speed of 12Mbits/s.

DP V0

DP V0 is the foundation for Profibus and was created by optimizing Profibus FMS to support fast data exchange. It is primarily used for process data exchange with the master cyclically reads input values and writes output values to slaves. In addition, DP V0 also has diagnostic reporting capability. [2] [4] [18] [32]

DP V1

Acyclic data exchange has been added on to DP V0 for parameterizing devices during operation and alarm acknowledgement. The acyclic data exchange runs parallel with cyclic data exchange, but with lower priority. [2] [4] [18]

DP V2

With the fundamental of the previous 2 versions, DP V2 extends its functionality to slave-slave communication without involving a master in isochronous mode. Isochronous mode synchronizes the device to achieve a data exchange in an exact time-slot pattern. [2] [4] [18]

Bus Termination

Bus termination avoids the reflections which disturbs the signal of the data communication. The bus termination is more essential while dealing with high baud rates and long cables. **Figure 8** is the comparison of the Profibus signal received by a slave with and without termination respectively. [2]

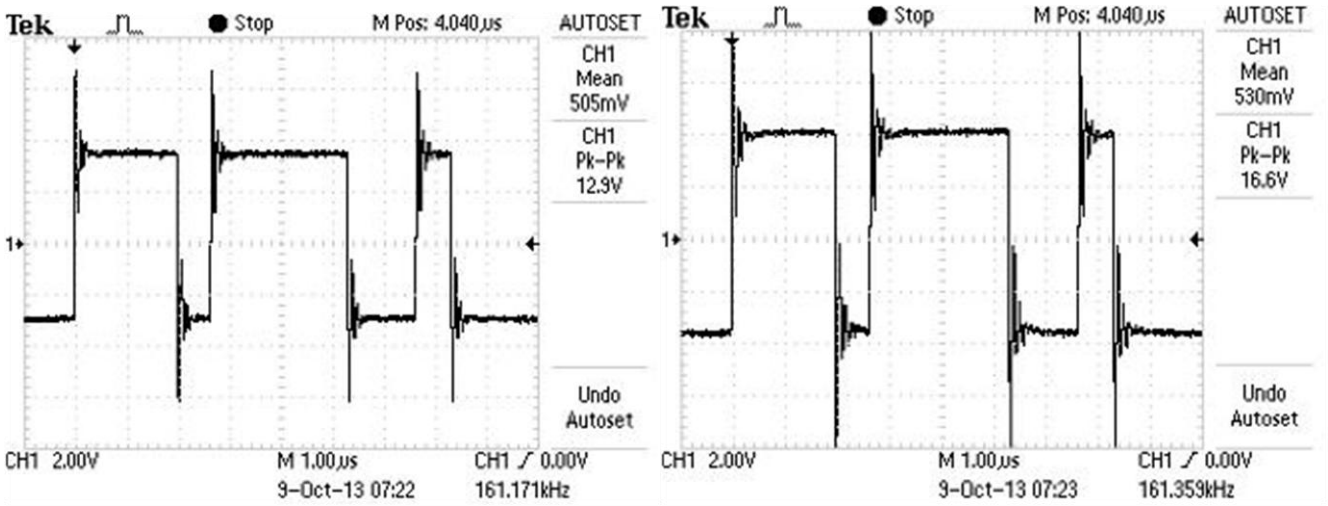


Figure 8: Comparison of Profibus signal with (left) and without (right) bus termination

The bus termination should be activated at each end of every bus segment. Profibus DP bus termination is powered at 5V to provide an idle level when no data is transferring. The bus termination function could be included within a connector or a device, whereas some devices require external bus termination. However, double bus termination on one end would cause corruption of the signals and network malfunction. [2]

Profibus DP Termination

The Profibus DP termination circuit is shown in **Figure 9**. [24]

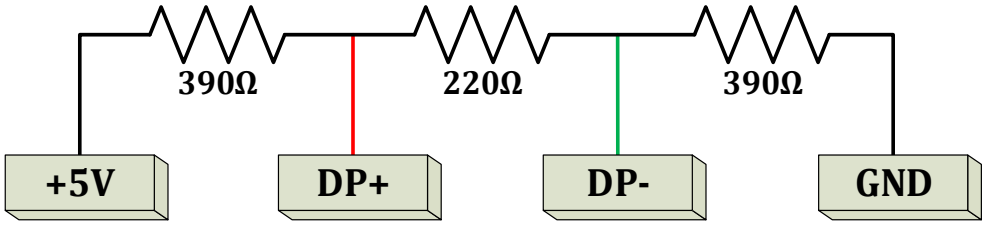


Figure 9: Profibus DP signal termination circuit [3]

Profibus PA Termination

The Profibus PA termination should be distinguished from Profibus DP termination and the circuit is shown in **Figure 10**. [19]

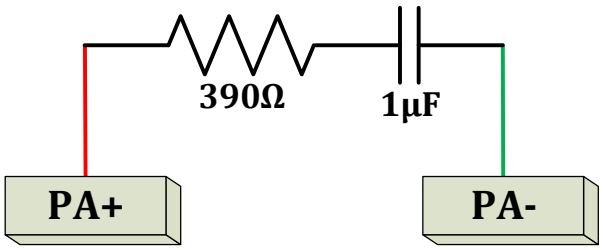


Figure 10: Profibus PA signal termination circuit [3]

Profibus Connectors

DP Connector

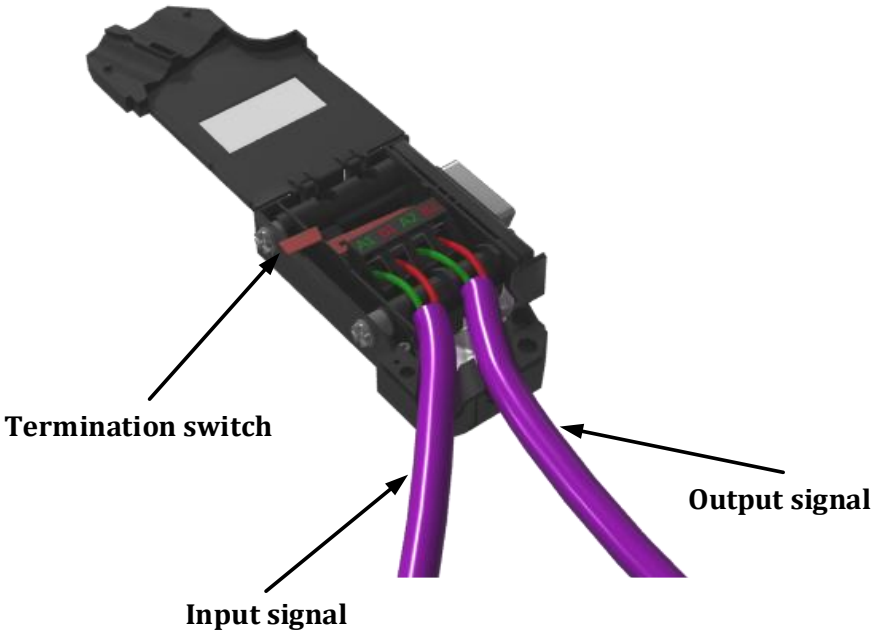


Figure 11: Profibus DP connector structure

The most common Profibus DP connector has a standard Sub-D9 connection and is able to loop the Profibus signal through multiple devices. It usually integrates a termination switch to manually set the termination. When the termination switch is on, the signal will be terminated and will no longer be sent to the next device. **Figure 11** shows the internal structure of a standard Profibus DP connector.

PA Connector

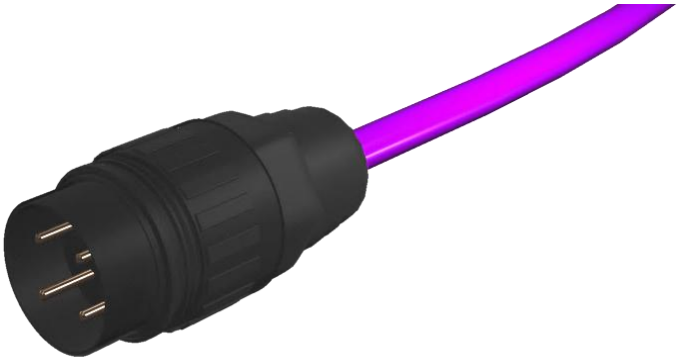


Figure 12: Profibus PA M12 connector

The M12 connector is the most common Profibus PA connector (**Figure 12**). With Manchester Bus Powered (MBP) technology, the Profibus cable carries both fieldbus data and power to supply PA instruments to reduce the risk in the explosive area.

Cabling

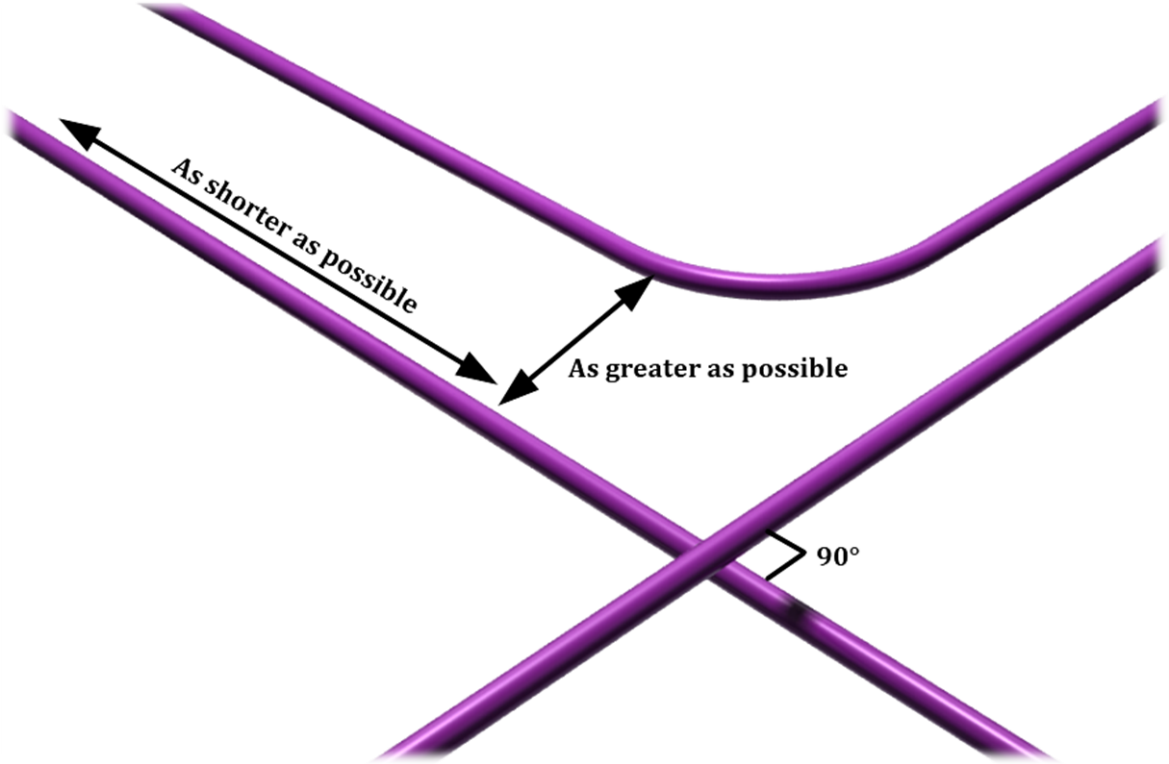


Figure 13: Profibus cable interference illustration [3]

In order to reduce the crosstalk interference, the spacing between Profibus cables needs to be as large as possible and the parallel portion of the cable path needs to be as short as possible. Whenever 2 cables need to cross over, the angle between them needs to be 90°. **Figure 13** demonstrates the above situations. The safety temperature range for Profibus cable is generally between -40°C to 60°C. [3] [6]

Profibus Configuration Software

The configurator is required to specify which slave devices should be seen by the master on the network and also prepare the information that needs to be transferred to each slave during the startup phase. All this information is typically from a configuration database file which is created from the Profibus configuration software. The Profibus configuration software would cover the following functions:

- Import GSD files and store the information in the hardware catalogue;
- Assign Profibus masters and slaves with their addresses;
- Allow master and slave data exchange;
- Specify the operating mode and bus baud rate;
- Generate database file for the master. [2]

Profibus Address

The available Profibus address range is from 0 to 126. Address number 0 is used by the configuration tool, address numbers 1 and 2 are normally assigned to the HMI device and the master respectively. Address 126 is a default address for a slave to allow address to be set via configuration software. Address 127 is a broadcast address and only visible for bus monitor if there is. Some slaves may block some particular addresses and some masters may have a limit. The address for each device or station must be unique within the network and the address sometimes can only be set manually on a device. [2]

Master and Slave

The core of the Profibus communication is master/slave method, where a master controls/monitors the entire system by sending requests to its slaves and collecting information from them. [85]

Master

Master Class 1

A C1 master is typically a PLC which performs data exchange with the slaves. It sets the baud rate and the slaves automatically detect the baud rate. During a DP V1 data exchange, the acyclical connection between the C1 master and slaves is automatically established, based on the foundation of cyclical connection. Profibus DP network only allows one C1 master. [1] [5]

Master Class 2

A C2 master does not hold a Profibus address and is normally a visualization system used for diagnosis, configuration and startup. It does not perform cyclical data exchange with slaves. The C2 master uses acyclic communication with slaves that are established by the initiate service. An acyclic connection allows read and write access to the slaves. Multiple C2 masters can be included in a Profibus network at the same time and the number of the maximum C2 masters allowed is determined by the slaves. [1] [5]

Slave

A slave is a peripheral device or passive station that responds to master requests and acknowledge messages. A slave does not grant bus control rights and can only have one master. All slaves in a network have the same priority and the communication only originates from the master. The GSD file of the slave defines the slave characteristic and I/O information for the master. [1] [45]

GSD File

GSD stands for General Station Description and is an ASCII text file that identifies the slave station. When a vendor develops a slave device, a GSD file must be developed as well which includes the complete description of the slave device such as baud rates supported, available input/output data configuration, etc. Those GSD files can be downloaded from the manufacturer's website or from the Profibus GSD library. Moreover, the GSD file can be read from the Notepad if necessary. **Figure 14** shows the GSD file download web page for Endress+Hauser products. [1] [2]

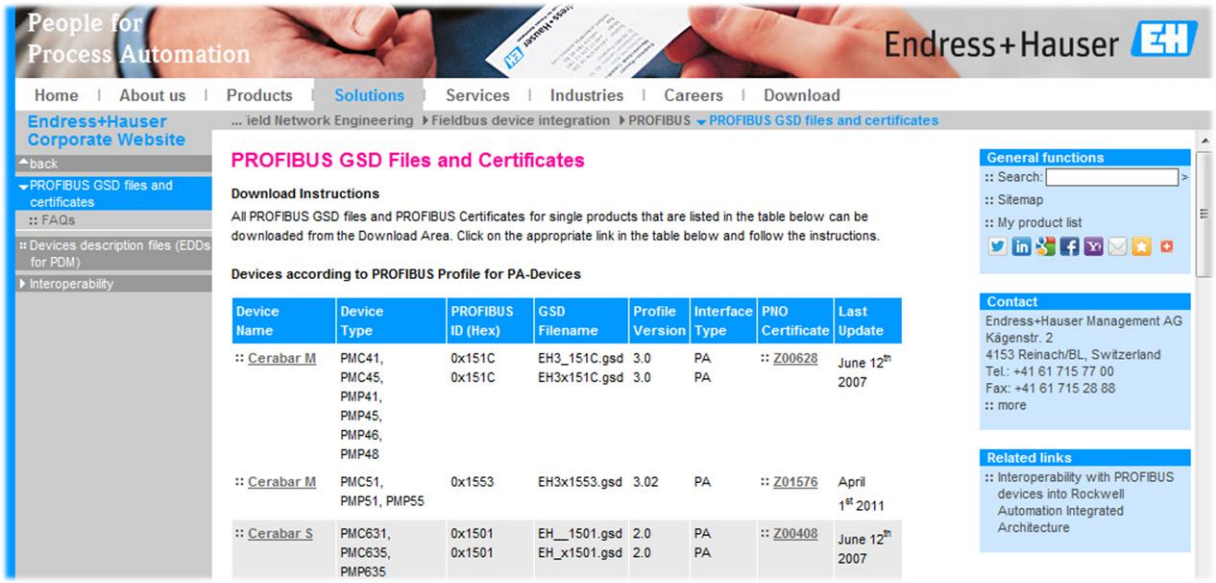


Figure 14: GSD files of instruments from Endress+Hauser website

Slave Modules

Slave modules are the data exchange schemes and structure defined in the GSD file for each device. A single GSD file can have multiple slave modules and they may or may not to be activated at the same time. Sometimes a slave device can only have one slave module at a time or must have at least one slave module to run depending on the manufacturers. When activating a slave modules, a certain length of peripheral input/output addresses will be added to the master controller for data exchange. [22]

Telegram

A telegram is a sequence of data which transfer between master and slave for data communication. In DP V0 protocol, the master sends a request telegram to the slave and expects to get a response telegram within a certain time. The slave, on the other hand, will reply a response telegram if it has successfully received the request telegram. [47] [48]

Figure 15 is a typical telegram for request and response. The Start delimiter (SD) is used to distinguish between request and response telegrams

Figure 15: Profibus telegram structure [47]

- **SD**: Start Delimiter.
- **ADR**: Slave Address.
- **TYP**: PDU type.
- **PDU**: Protocol Data Unit.
- **BCC**: Block Check Character. [47]

Data Exchange

Figure 16 demonstrates both cyclic and acyclic read data exchange between 1 master and 2 slaves.

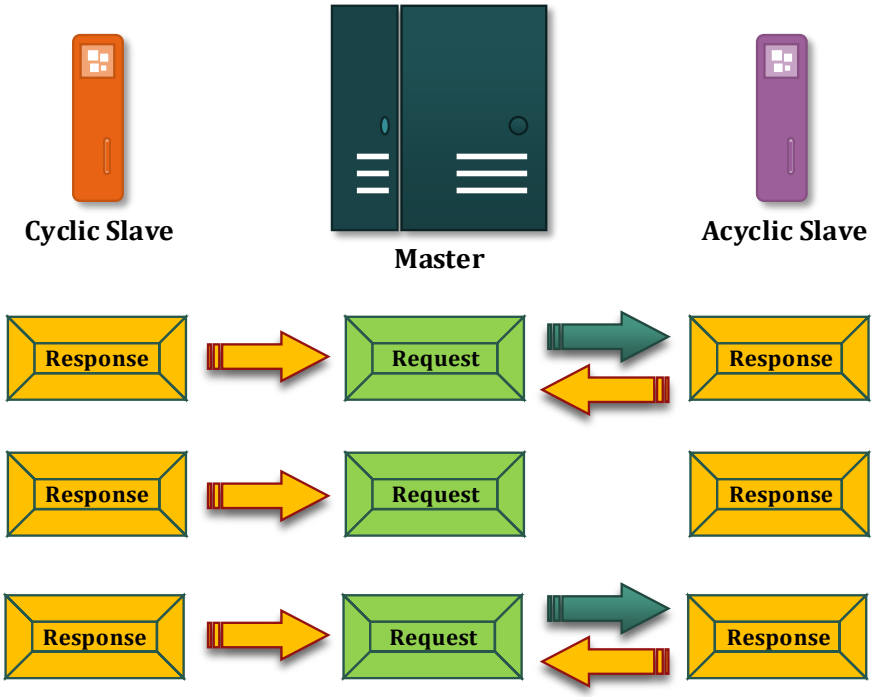


Figure 16: Comparison of cyclic and acyclic data exchange

Cyclic Exchange

The cyclic data exchange is mainly designed for continuously controlling or monitoring peripheral devices at a fixed certain rate such as process data exchange. [47]

Acyclic Exchange

The acyclic data exchange is primarily used for diagnosis and startup purpose so the data exchange event can occur upon request. Typical acyclic data exchange involves parameter data exchange. [47]

Process Data

The process data are time critical fast-transmitted data with the high dynamic property and actuality. They are exchanged cyclically between the master controller and the slaves. The process input data and the process output data are treated individually. This enables the user to specify the types of the applications without any confusions. For example, master controller can use process input data to read the actual values and write setpoints to the process output data. [47]

Control Word

The control word is widely used for drive communication and is usually included in the process data. The control word normally contains 16 bits with fixed function assignments. It is typically used to control the operation of a slave device.

Status Word

Similar to the control word, the status word collects information from the slave devices in a cyclical data exchange. The structure of the 16 bit status word is usually defined by the manufacturer.

Communication Processor

Figure 17: CP5611 Profibus communication processor PCI card [15]

The communication processor is used to provide a communication path between PC and PLC. In general, different communication protocols require different communication processors. The available communication processor for Profibus is a CP5611 Profibus PCI card (**Figure 17**). It supports local communication with up to 30 process units via Profibus. The CP5611 PCI card has a Sub-D9 connector which can be easily connected using a standard Profibus DP connector. [15]

Network Gateways

With the help of gateways, a Profibus network can be easily linked to other networks who have different protocols. **Table 1** lists some common gateways with their supported protocols. [32] [8]

Network 1	Network 2	Gateway
Profibus	Profibus	DP/DP coupler
Profibus DP	Profibus PA	DP/PA coupler
Profibus	Industrial Ethernet/Profinet	IE/PB link or IE/PB link PN IO
Profibus	Industrial wireless LAN	IWLAN/PB link PN IO
Profibus	AS-Interface	DP/AS-I link

Table 1: Network gateways for linking 2 different fieldbuses [32]

Figure 18 demonstrates the general structure of multi-fieldbus network built by various gateways in **Table 1**.

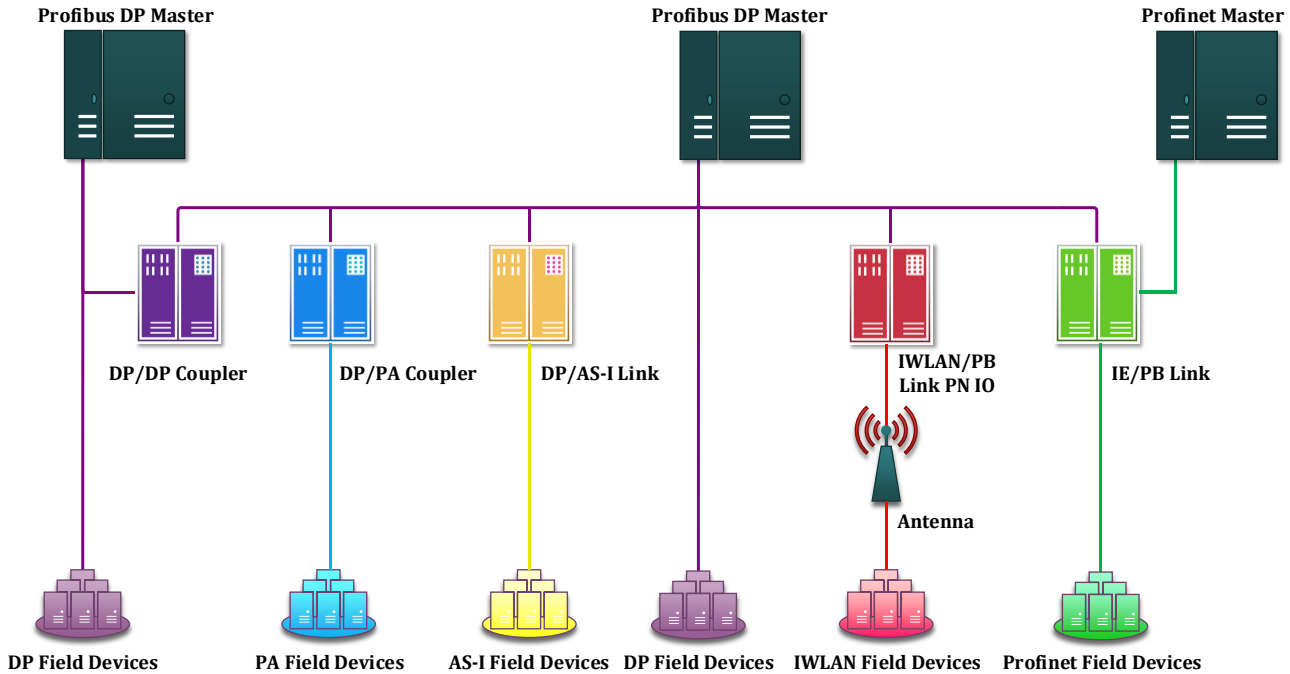


Figure 18: Multi-fieldbus network with gateways [32]

OPC

Figure 19: OPC logo [113]

OPC (**Figure 19**) stands for Object linking and embedding Process Control and is defined by OPC Foundation as a communication standard which represents over 300 automation companies. It is a manufacturer-independent software interface based on Component Object Model (COM) and Distributed Component Object Model (DCOM) technology. The foundation of OPC provides a standard for devices and applications from various manufacturers to access process data from each other. [18] [85]

OPC Network Components

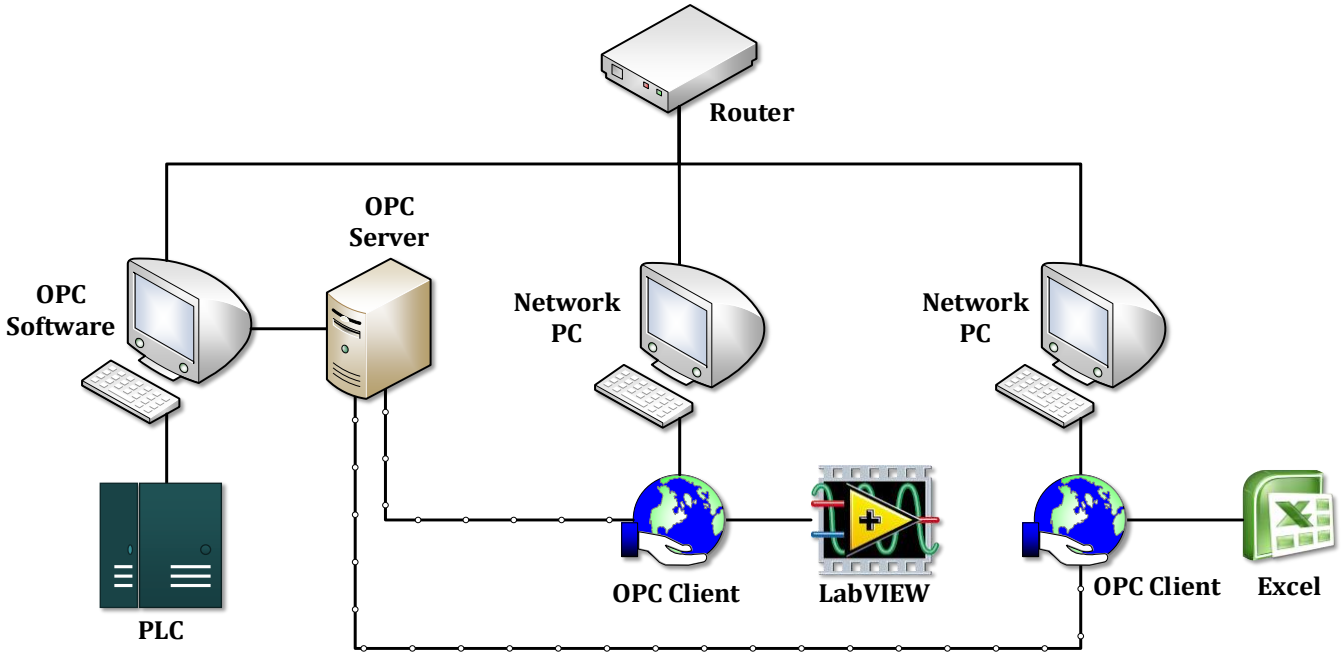


Figure 20: Generic OPC communication network

OPC Server

The OPC server is a program which sets up a standard software interface for applications of various manufacturers. It acts as an intermediate layer between process data handling application and process data access interface.

OPC Client

OPC client is an application which is able to access the process data from an OPC server. One OPC client can access process data from multiple OPC servers. **Figure 20** shows a standard OPC communication network where LabVIEW and Excel can access data from a PLC.

OPC Framework Components

COM

COM is a standard protocol for communication between objects on the same computer but in different applications. The server is able to recognize those objects and allows the clients to access them. [85]

DCOM

DCOM expands the capability of COM and allow clients to access objects from different applications on remote computers. [107]

OPC Specifications

WinCC supports 4 types of OPC specifications:

- OPC Data Access (OPC DA);
- OPC Alarm & Events (OPC A&E);
- OPC Historical Data Access (OPC HDA);
- OPC eXtensible Markup Language DA (OPC XML-DA). [85]

OPC Data Access (OPC DA)

The OPC DA specification defines an interface to read and write data in realtime. The WinCC OPC DA server makes its online tags available to other applications running within the same network. On the other hand, WinCC OPC DA client can also access the process tags of a PLC or application of other manufacturers. [85]

OPC Historical Data Access (OPC HDA)

The OPC HDA server provides read and write access to the data in a WinCC archive system for all OPC HDA clients. [85]

OPC Alarms & Events (OPC A&E)

OPC A&E specification defines an interface for event monitoring. An OPC A&E server is typically used on a WinCC server and behaves like a condition related event server to track events. All the OPC A&E clients can access the OPC A&E server. [85]

OPC eXtensible Markup Language DA (OPC XML-DA)

The OPC XML-DA is a standard that uses a platform-dependent protocol over the Internet to enable communication. This communication is implemented as a web service of Microsoft Internet Information Server (IIS). The OPC XML-DA server supports OPC XML-DA clients with process data access when using the Internet via HTTP. [85]

Configuration Software

TIA Portal

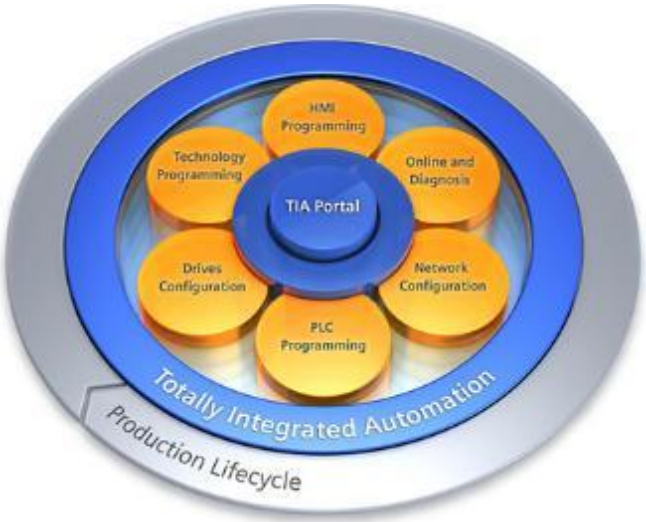


Figure 21: TIA Portal Logo [38]

TIA Portal (**Figure 21**) is a modern integrated automation configuration software delivered by Siemens and is actually a new version of STEP7. The purpose of TIA portal is to achieve TIA (Totally Integrated Automation) concept in order to shorten commissioning time. It is capable of writing PLC programs, HMI designs, configuring peripheral devices and developing SCADA systems. TIA Portal supports all 5 standard PLC programming languages and is able to convert between different programming languages as well. [38]

(Refer to Appendix V S7-300 PLC and Repeater Configuration Instructions for TIA Portal configuration)

WinCC flexible

Figure 22: WinCC flexible startup screen [59]

WinCC flexible (**Figure 22**) is fundamentally a HMI device configuration software which governs the communication between a PLC and HMI device by establishing the connection between external tags and variables in the PLC. It is also equipped with a range of objects, controls, functions and VBScript to create dynamic HMI screens for operation visualizations. As TIA Portal is now able to configure both PLCs and HMI devices, it is unnecessary to introduce WinCC flexible. However, the compatibility of WinCC flexible is better than TIA Portal with some preset functions and it can also be considered as a backup solution. [59] [67]

(Refer to Appendix XI WinCC flexible Configuration Instructions for WinCC flexible configuration)

WinCC

Figure 23: WinCC startup screen [86]

WinCC (**Figure 23**) stands for Windows Control Center. It is an industry and technology system for solving control-related tasks in production and process automation. WinCC allows users to create an automation system with an HMI interface for visualization and fast commissioning by means of its GUI. WinCC consists of 2 parts: the Configuration System (CS) and Runtime (RT). The core of CS is WinCC Explorer which acts as a project planner that contains the entire project structure to allow users to retrieve different editors. The RT, on the other hand, allows communication with the automation system and performs process operation and observation. [86] [89]

(Refer to Appendix XIII WinCC Configuration Instructions for WinCC configuration)

LabVIEW

Figure 24: LabVIEW logo [2]

LabVIEW (**Figure 24**) stands for Laboratory Virtual Instrumentation Engineering Workbench, a platform and development environment for a graphical programming language developed by National Instruments. LabVIEW provides engineers and scientists with a proven highly productive development environment to design control systems and take measurements. LabVIEW supports various types of communication protocols and hardware integration and also allows users to create modern looking HMI interfaces rapidly. It is able to communicate with other automation systems over a range of communication toolboxes to achieve the maximum performance in the industry.

CHAPTER 5: WINCC SYSTEM

WinCC Components

Tag System

The internal tags in WinCC are calculated and simulated within WinCC. The external tag is the connection between WinCC and PLCs. Each external tag represents a variable in a PLC such as a memory location, input or output. During Runtime, the process values of external tags are monitored and controlled by WinCC via communication driver channels. [91]

(Refer to Appendix XIII WinCC Configuration Instructions for tag configuration)

Graphics System

The graphics designer in WinCC is a vector based drawing program for creating user defined process screens. It provides a range of basic shapes and windows objects such as buttons, tick boxes and so on. It also contains a graphics library with lots of preset pictures that are ready to use. Other than those, the graphic designer also integrates programming languages such as ANSI-C and VBScript to expand the flexibility of the HMI interface. [86]

(Refer to Appendix XIII WinCC Configuration Instructions for graphics configuration)

Alarm System

The alarm system allows users to define customized alarm conditions for the detection of critical situations. Both digital and analog alarms are available, with user defined alarm types and classes to enhance the process monitoring system. Configuration could be done to display a user defined message with other customized information when an alarm triggers and the acknowledgement can be executed from various sources. [100]

(Refer to Appendix XIII WinCC Configuration Instructions for alarm configuration)

Archive System

The logging system is a dispensable part of WinCC system. It archives all the tag values over time and alarm messages, then stores them in a WinCC database. The user can configure the start and stop time of the logging and also compress the data in miscellaneous ways to minimize the storage space. [103]

(Refer to Appendix XIII WinCC Configuration Instructions for archive configuration)

WinCC Communication Structure

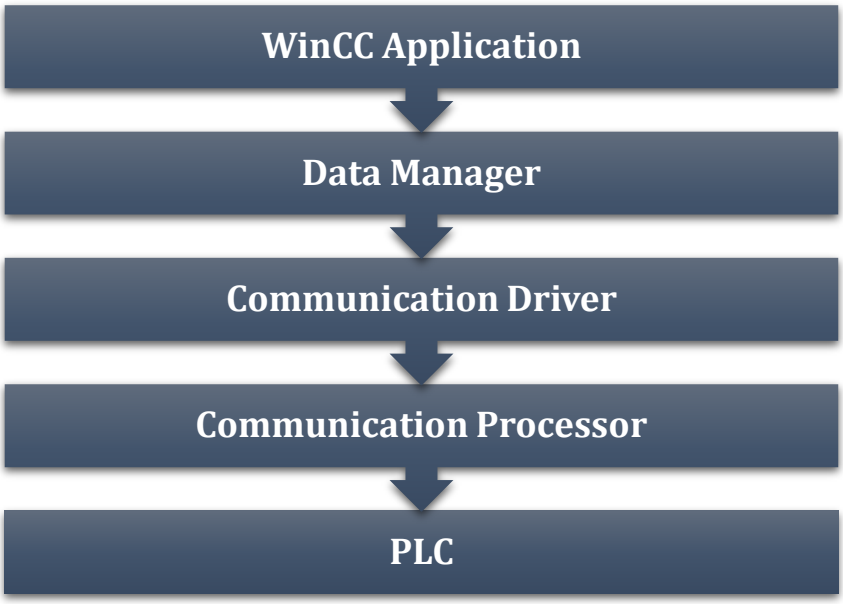


Figure 25: WinCC communication structure [85]

Data Manager

Data manager is the WinCC data handle mechanism that is not visible to users. All WinCC Runtime applications such as tag logging and alarm logging must request the data from the data manager as WinCC tags. [85]

Communication Driver

Communication drivers in WinCC establish the communication between data manager and PLC via different communication channels. Each communication driver available represents a Dynamic Link Library (DLL) that communicates with data manager through Application Programming Interface (API). [85]

Communication Channel

A communication driver may contain one or more communication channels and each of them may have their own communication protocol. [85]

Communication Processor

The communication processor provides an interface for a PC to recognize certain communication methods and allows data exchange with an automation system. [85]

Script

WinCC integrates VBScript and ANSI-C script to provide a flexible programming environment and also improve the efficiency of the configuration. Even though both scripts can satisfy most of the applications, they do have their strengths and weaknesses. VBScript is suitable when dealing with object properties, databases and data exchange between WinCC and Microsoft Office. ANSI-C is more convenient for accessing other applications in the Windows platform via an API. **Table 2** shows the comparison of the operation speed of VBScript and ANSI-C based on a PC with Pentium 4 CPU, 2.5GHz, and 512MB RAM. [29]

Actions	VBScript	ANSI-C
Set colour for 1000 objects	220ms	1900ms
Read values from 1000 internal tags	60ms	170ms
Set values for 200 I/O fields	920ms	500ms
Perform 100000 calculations	280ms	70ms

Table 2: Strength and weakness of ANSI-C and VBScript [29]

(Refer to Appendix XIII WinCC Configuration Instructions for more information about script specifications)

Siemens OPC DAAutomation 2.0

Siemens OPC DAAutomation 2.0 is an add-on which provides a dynamic link library for easy accessing WinCC tags from other applications. It establishes an interface for other Windows applications to communicate with the automation system via VBA and VC++. It is automatically integrated into Excel while installing WinCC and it can be activated from the Visual Basic references window. [29] [107] [113]

(Refer to Appendix XIII WinCC Configuration Instructions for Siemens OPC DAAuomation 2.0 setup)

CHAPTER 6: DEVICES

S7-300 PLC

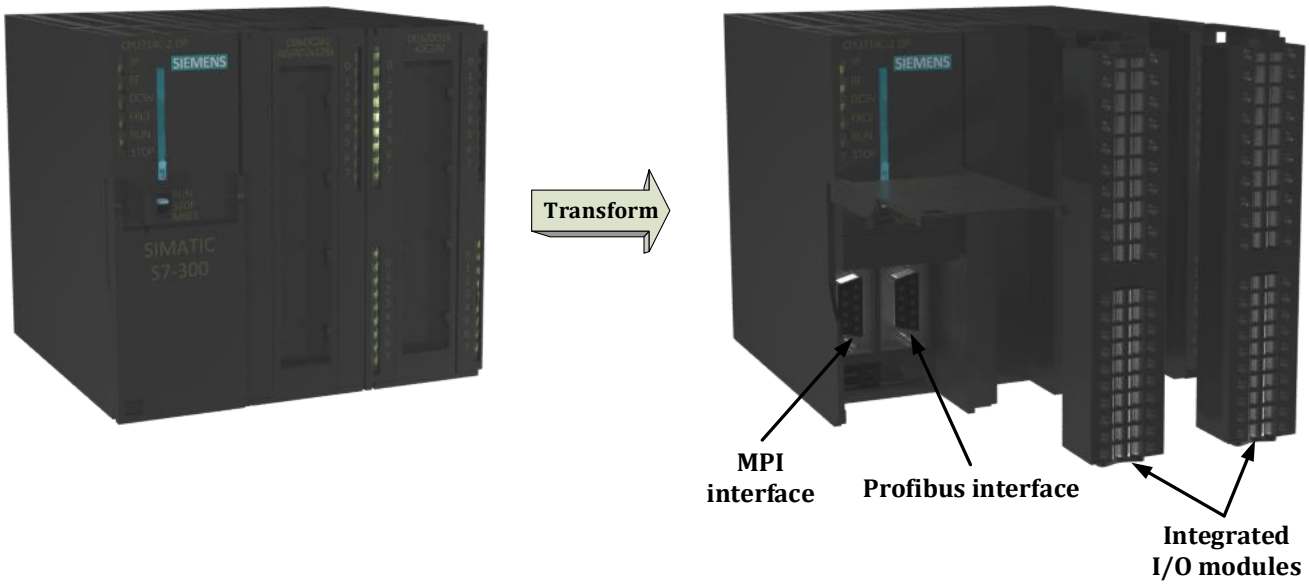


Figure 26: Siemens S7-300 PLC structure

The available PLC model is a Siemens CPU314C-2DP S7-300 PLC with MPI and Profibus interfaces and contains two integrated modules for analog and digital I/O (**Figure 26**). [31]

The S7-300 PLC is completely compatible with TIA Portal or even STEP 7. User can write PLC program in various standard languages in the configuration software and download to the PLC to perform the task.

(Refer to Appendix V S7-300 PLC and Repeater Configuration Instructions for module and configuration information)

TP 177B 6” HMI Touch Panel

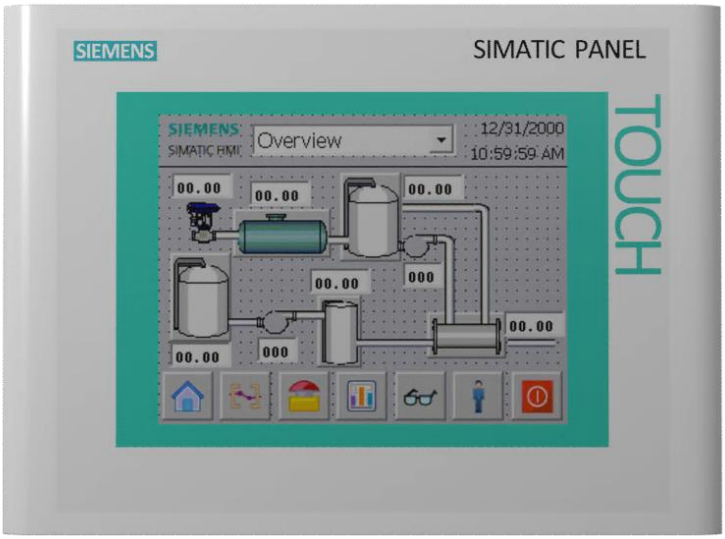


Figure 27: TP 177B 6" HMI touch panel

The Human Machine Interface (HMI) device provides maximum transparency for the operators who work in a complex process environment and highly increases the machine and plant functionalities. A maximum of 4 HMI touch panels are allowed on the same bus over an Ethernet communication with 1 server and 3 clients. The HMI system essentially has the following features:

- **Process visualization**: The process system is visualized on the HMI device and the objects and values on the HMI screen update dynamically;
- **Operator process control**: Operator with authorization is able to control the system via the HMI device by means of the GUI;
- **Trending and alarm**: The historical process values and alarm messages can be displayed on the screen of the HMI device;
- **Process value and alarm logging**: HMI device generally has the capability of exporting data for analysis purpose. [60]

An HMI tag represents an image of a defined memory location in the PLC and they are dynamically linked. As soon as an HMI device is bound to the PLC, the HMI device is able to access the defined variables in the PLC and displays them on the screen so that the operator can enter values on the HMI device to remote control the PLC behaviour. [59]

TP 177B 6” is compatible with both WinCC flexible and TIA Portal for configuration and commissioning. However, an update of operating system is required while changing the configuration software. [70]

(*Refer to Appendix IX TP 177B 6” System Configuration Instructions for module and configuration information*)

EASY719

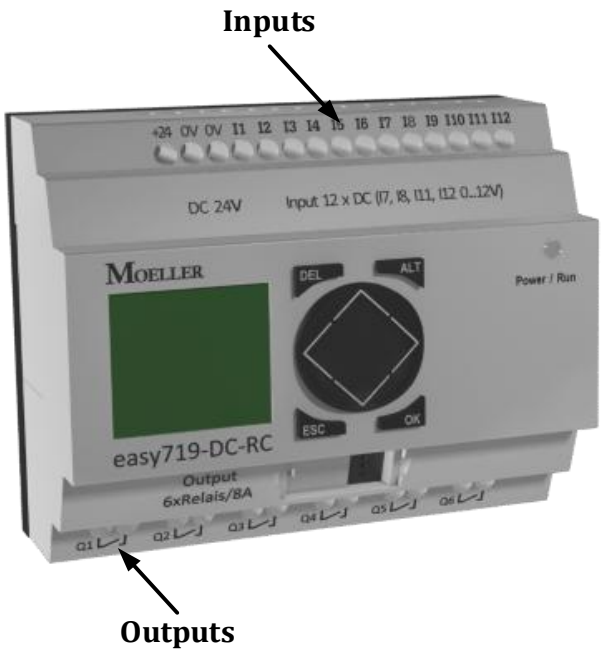


Figure 28: EASY719 controller

EASY719 (**Figure 28)** is a robust smart relay controller with logic functions, timers, counters and time-switch functions. It is capable of performing many kinds of complex control systems such as plant construction.

EASY719 uses logic ladder diagrams to connect circuits. The circuit can be entered either from operating buttons or by programs transferring from EASYSOFT. EASY719 allows users to perform the following functions:

- Detect digital inputs;
- Set output relays;
- Count high speed pulses;
- Frequency measurements;
- Analog inputs;
- Text display. [41]

(Refer to Appendix VI EASY719 & EASY204-DP Configuration Instructions for module and configuration information)

EASY204-DP Gateway

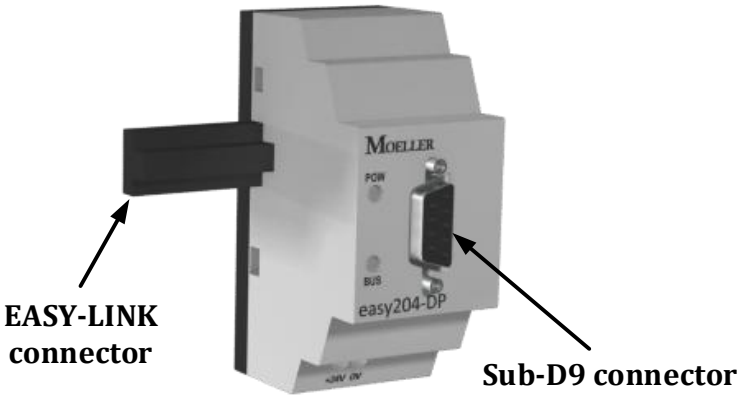


Figure 29: EASY204-DP Profibus gateway

EASY204-DP (**Figure 29**) is a Profibus gateway which provides a communication path between an EASY719 and Profibus master. The gateway is connected to an EASY719 as an expansion device with EASY-LINK connector at the backplane. EASY719 with Profibus DP gateway works as a slave in the Profibus network. [42]

(Refer to Appendix VI EASY719 & EASY204-DP Configuration Instructions for module and configuration information)

EASY-LINK

EASY-LINK is a bus connector used to connect expansion units such as a gateway or analog output expansion units to the basic controller unit. EASY-LINK recognizes expansion device by cyclically checking whether any data is sent from the expansion device. [42]

EASY719 can be controlled via different fieldbus systems with different gateways, **Table 3** shows some available gateways associated with different fieldbus systems. [42]

Fieldbuses	Gateways
Profibus DP	EASY204-DP
ASI	EASY205-ASI
CANopen	EASY 221-CO
DeviceNet	EASY 222-DN

Table 3: Available EASY719 gateways for different fieldbus systems [42]

EASYSOFT

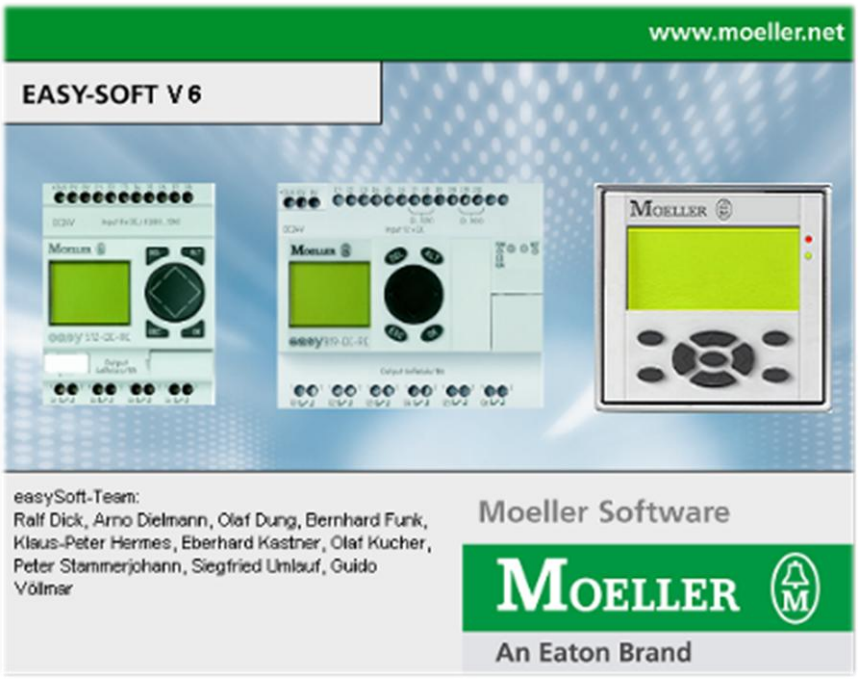


Figure 30: EASYSOFT logo

EASYSOFT (**Figure 30**) is a PC software that provides a comfortable user interface allowing users to create, store and simulate ladder circuit diagrams. Nearly all the functions could be performed in EASYSOFT and the completed circuit diagram could be downloaded to EASY719 using an Easy-PC-CAB cable over RS232 communication.. EASYSOFT is able to perform the following operations:

- Design circuit diagram;
- Set up function parameters;
- Download and upload program;
- Online monitoring;
- Circuit simulation.

(*Refer to Appendix VI EASY719 & EASY204-DP Configuration Instructions for EASYSOFT configuration*)

Slave Modules

Figure 31 shows all the available slave modules from the EASY204-DP GSD file.

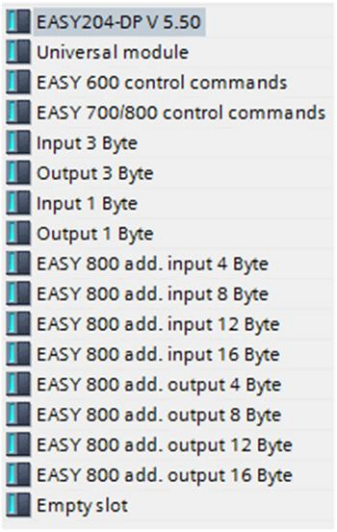


Figure 31: Slave modules of EASY204-DP in TIA Portal

Figure 32 is the overall structure of all the compatible slave modules for the EASY719 controller.

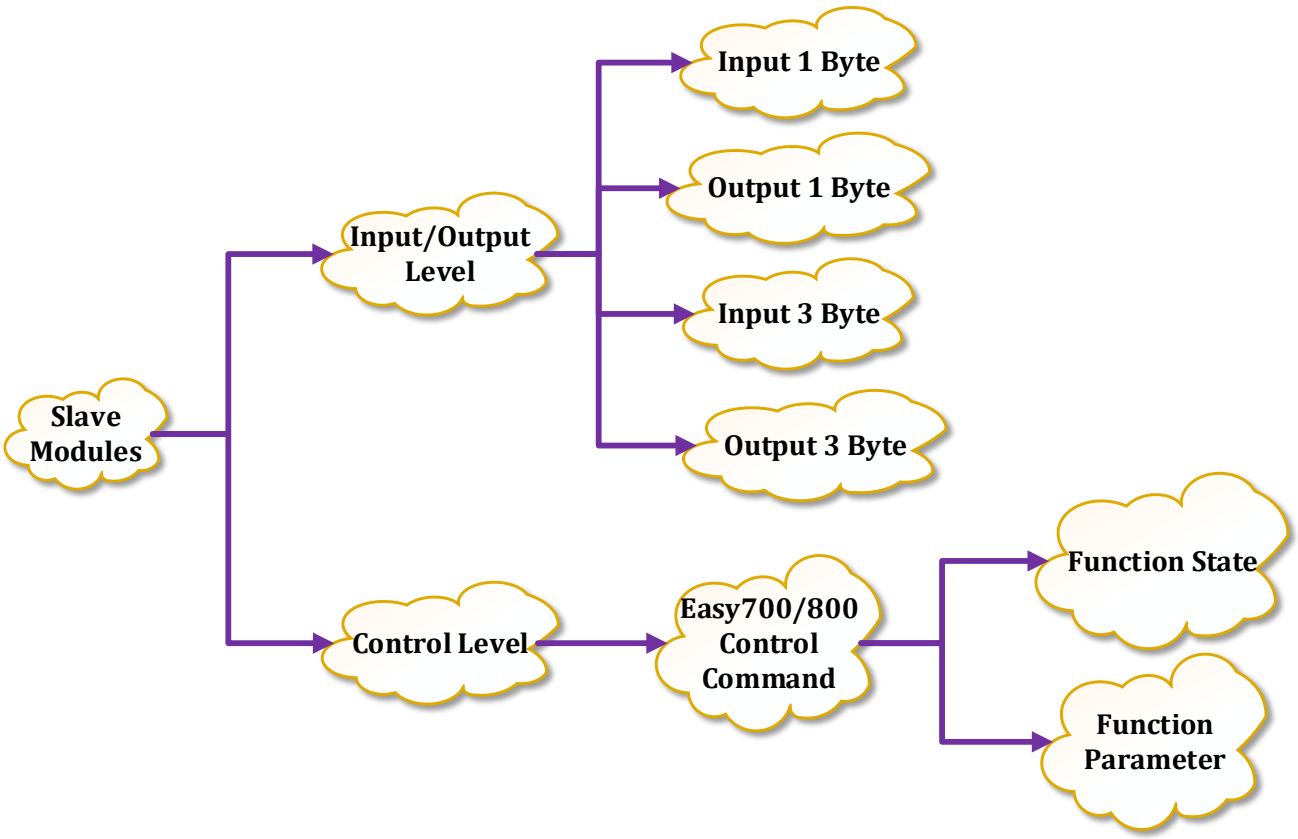


Figure 32: Overall structure of all the EASY719 compatible slave modules

Input/output Level (Cyclical)

The **Input/Output 1 Byte** slave modules only have the functionality to read and write state values, whereas **Input/Output 3 Byte** slave modules can also read and write operating modes and write to more outputs. Generally, **Input/Output 1 Byte** and **Input/Output 3 Byte** slave modules are used to read inputs (R) and write outputs (S) to the expansion devices. However, a simple program could be developed to link the expansion device I/O to the basic unit I/O to indirectly read and write states to EASY719.

Control Level (Acyclical)

EASY 700/800 control command slave module is the designed slave module for data exchange between EASY719 basic unit and a PLC. It can be divided into 2 types. The first type is used to read/write logical states or monitor analog inputs and the second type is used to access function block parameters. This slave module can be present with any other modules.

(Refer to Appendix VI EASY719 & EASY204-DP Configuration Instructions for Profibus data exchange configuration)

MOVITRAC B

MOVITRAC B is a variable speed drive from SEW. The available model is a single phase drive and it is controlling a SEW single phase induction motor (**Figure 33**).

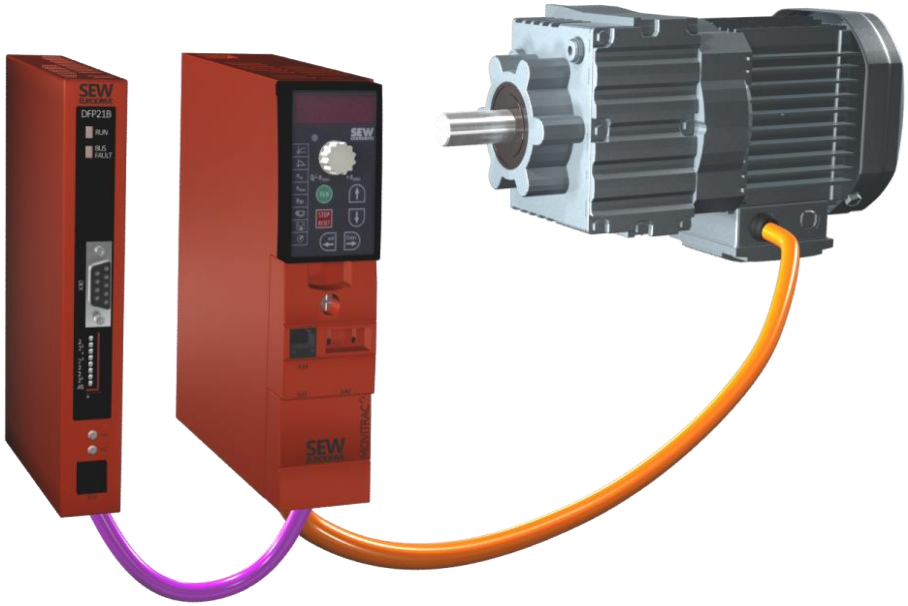


Figure 33: DFP21B, MOVITRAC B and the connected single phase motor

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for module and configuration information)

Specification

Table 4 shows the specification of the MOVITRAC B and the motor connected.

Specification	Description
Model	MC07B0003-2B1-4-00
Line connection	1 phase, 200 – 240V at 50/60Hz
Nominal output current	1.7A AC
Speed range	-3000 – 3000RPM
Motor model	SEW R17 DR63M4
Motor power	0.18kW

Table 4: MOVITRAC B and motor specifications [43]

Hardware Modules

Figure 34: MOVITRAC B structure

MOVITRAC B comes with 2 pluggable module slots for the keypad module (FBG11B) and a communication module (FSC11B) to extend its capabilities and visualization as shown in **Figure 34**.

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for module and configuration information)

FBG11B Keypad Module

Figure 35: FBG11B keypad module

The FBG11B keypad module (**Figure 35**) is used for diagnostics and startup and is also able to change the parameters without a PC or PLC. It provides an extremely flexible control and monitoring interface. The FBG11B module can be plugged and unplugged from MOVITRAC B during operation. Its main functions are as follows:

- Display process values, customized information and status;
- Setting parameters with choice of short or long menu;
- Startup motor;
- Local manual control;
- Data backup and transfer. [43]

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for module and configuration information)

FSC11B Communication Module

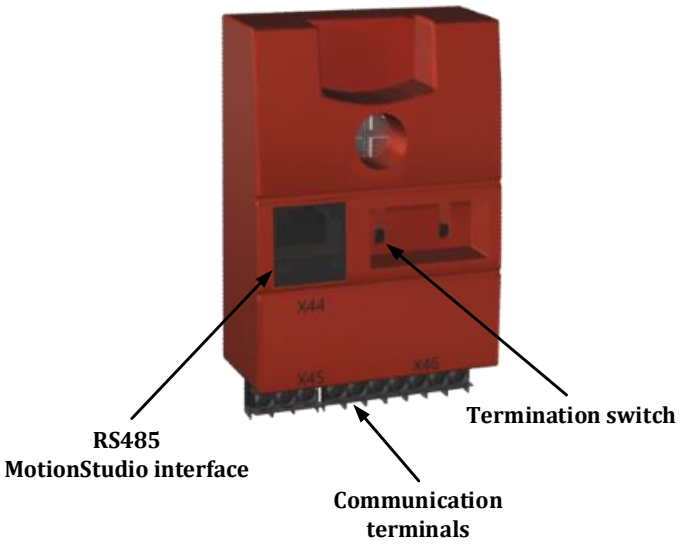


Figure 36: FSC11B communication module

The FSC11B communication module (**Figure 36**) provides a path for MOVITRAC B to communicate with a gateway, PC or even another MOVITRAC B. It supports RS485, SBus with MoviLink and CANopen protocols. Termination resistor S1 should only be turned on to terminate the bus signal at the end of the end of the network. [43]

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for module and configuration information)

Communication Adaptor

USB11A Adaptor

USB11A is an RS485 to USB adapter used to connect MOVITRAC B/DFP21B to a PC via the communication module. It has an RJ10 socket for serial RS485 interface and supports both USB 1.1 and USB 2.0. MotionStudio can perform data exchange between MOVITRAC B/gateway and a PC using this adapter. [43]

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for connection information)

UWS21B Adaptor

UWS21B is an RS485 to RS232 adapter and serves exactly the same functionality of USB11A. [43]

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for connection information)

DFP21B Gateway

The DFP21B gateway provides a Profibus interface for MOVITRAC B and it consists of a UOH11B housing with additional diagnostic functions and a Profibus interface card (**Figure 37**). A FSC11B communication module is used to connect the gateway and MOVITRAC B.

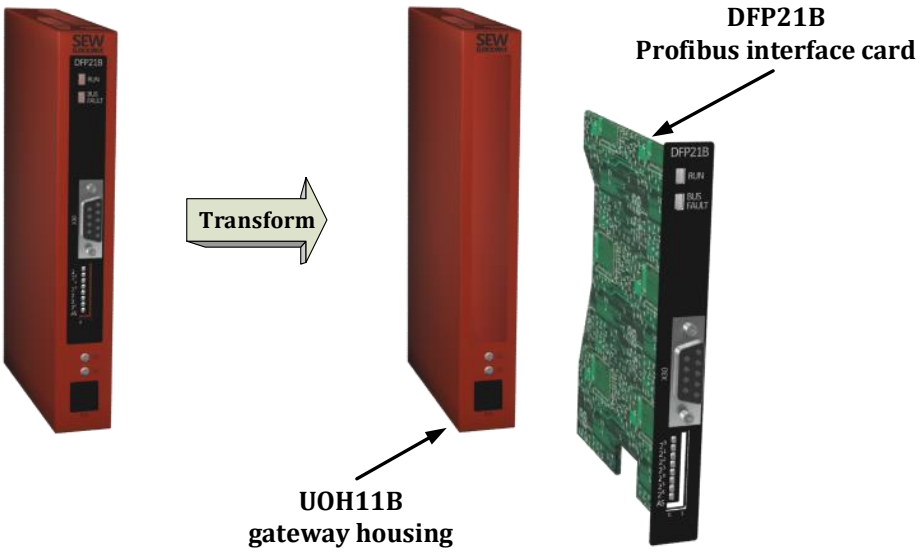


Figure 37: DFP21B gateway structure

The DFP21B gateway is capable of translating between Profibus DP protocol and SEW SBus protocol to establish a communication path between a Profibus master and MOVITRAC B. One gateway can support data exchange up to 8 MOVITRAC B at the same time as shown in **Figure 38**.

Figure 38: DFP21B with 8 MOVITRAC B with SBus communication

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for module and configuration information)

UOH11B Gateway Housing

Figure 39: UOH11B gateway housing

The UOH11B housing (**Figure 39**) does not only provide a mounting area for the DFP21B card, but also integrates some important diagnostic capabilities for SBus communication such as an RJ10 socket for MotionStudio interface. It also has 2 LEDs to display the communication status between the gateway and MOVITRAC B.

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for module and configuration information)

Gateways for Supported Fieldbus Systems

SEW also has gateways for other fieldbuses, **Table 5** lists all the supported protocols with their gateway models. [43]

Fieldbuses	Gateways
Profibus	DFP21B
Profinet	DFE32B
DeviceNet	DFD11B
EtherCAT	DFE24B
EtherNet/IP + Modbus/TCP	DFE33B
Profibus/PROFIsafe	DFS11B
Profinet/PROFIsafe	DFS21B
Interbus	UFI11A

Table 5: Available gateways for various fieldbus systems [43]

Communication Protocols

MoviLink

MoviLink is a SEW specific profile. It implements the data exchange between a master and multiple SEW drives. The master could be a PLC or PC. MoviLink does not depend on the transmission medium, so its capabilities can be used on serial or fieldbus data transmission. [46]

SBus

The SBus is a type of CAN bus based on CAN specification 2.0. All the MoviLink services are supported by the SBus. MOVITRAC B allows digital access to all the parameters via SBus at very high speed. On an SBus network, every unit including the gateway needs to have a different SBus address from 0 to 64.

The maximum cable length depends on the selected baud rate of SBus:

- 250 kBaud: 160m;
- 500 kBaud: 80m;
- 1000 kBaud: 40m.

A 2ms cycle time is required for each MOVITRAC B over SBus network which means 8 MOVITRAC B on SBus would take 16ms to update the process data. [46]

MOVITOOLS MotionStudio

Figure 40: MotionStudio logo [44]

MOVITOOLS MotionStudio (**Figure 40**) is a configuration tool for SEW products. With MotionStudio, user can establish communication with multiple units simultaneously and execute functions with consistency. It provides a neat interface to view all the available parameters for the drive and allow userS to make changes online. MotionStudio offers the following functions:

- Parameterization;
- Startup;
- Visualization and diagnostics;
- IPOS Programming;
- Scope;
- Data management. [44]

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for MCT 10 configuration)

Slave Modules

Figure 41 shows all the available slave modules from DFP21B GSD file where as **Figure 42** illustrates the overall structure of the available slave modules.

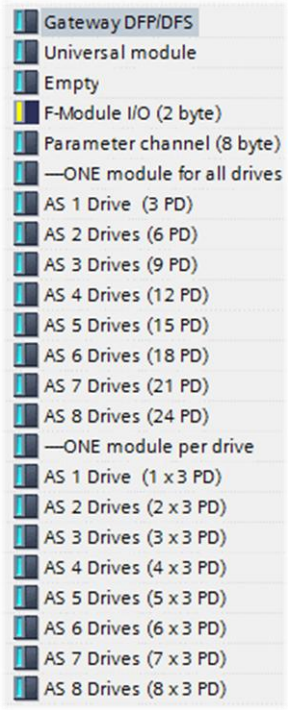


Figure 41: Slave modules of DFP21B in TIA Portal

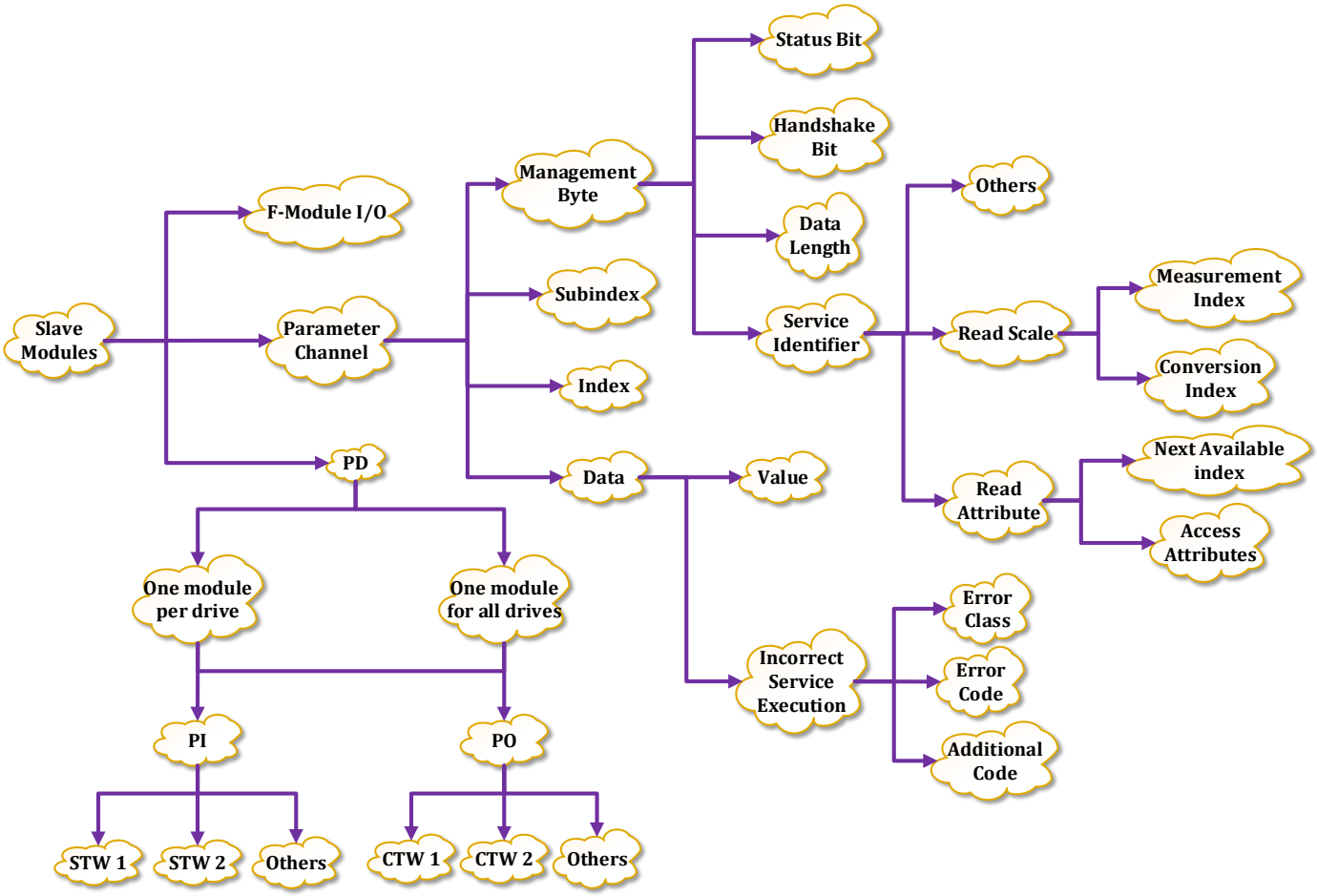


Figure 42: Overall structure of DFP21B slave modules

F-Module I/O

This module is used for PLC with PROFIsafe capabilities only.

Parameter Channel (Acyclical)

The **Parameter channel** slave module is used to access parameter data from the gateway only.

AS Module (Cyclical)

There are two types of modules available for data exchange between master and gateway. They organize the data in slightly different ways, but there are no critical differences between them.

One Module for All Drives

This module is able to read and write data to all the available drives in the network with a single long data.

One Module Per Drive

This module is able to read and write data to a single drive in the network with a short data.

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for Profibus data exchange configuration)

HVAC FC102

HVAC FC102 is a three phase HVAC drive from Danfoss vendor and is connected to a 3 phase motor as shown in **Figure 43**. It is capable of perform logic operations, smart logic state machine and cascade control.

Figure 43: HVAC FC102 and the connected 3 phase motor

(Refer to Appendix VIII HVAC FC102 Configuration Instructions for module and configuration information)

Specification

Table 6 gives the specification of HVAC 102 and the three phase motor connected.

Specification	Description
Model	FC-102P1K1
Line connection	3 phase, 380 – 480V at 50/60Hz
Nominal output current	2.7A AC
Speed range	-3000 – 3000RPM
Motor model	SEW SA57 R17 DR63S4
Motor power	0.12kW

Table 6: Specifications of HVAC FC102 and motor

Hardware Modules

HVAC FC102 comes with an LCP102 keypad module only. The Profibus interface card (MCA101) and the Sub-D9 connector module (MCF104) were purchased separately (**Figure 44**).

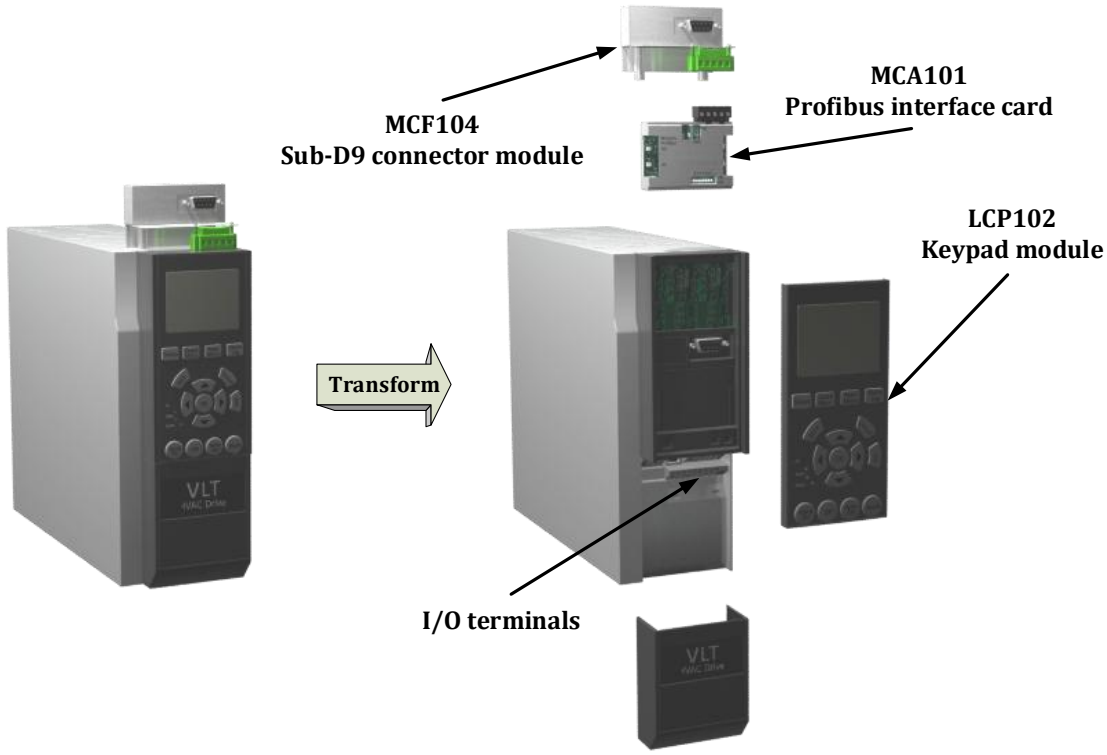


Figure 44: HVAC FC102 structure

LCP102 Keypad Module

Figure 45: LCP keypad module

The LCP102 keypad module (**Figure 45**) is used to set up the drive parameters and perform local manual control as basic functions. It also supports some advanced functions as follows:

- Save customized parameters in **Quick menu;**
- Configure the text display;
- Show help information for every parameter;
- Display trending of parameter on LCD screen.

(Refer to Appendix VIII HVAC FC102 Configuration Instructions for module and configuration information)

MCA101 Gateway

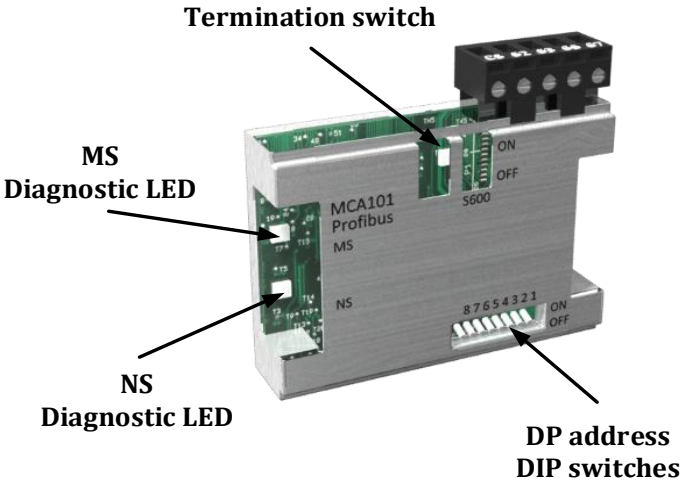


Figure 46: MCA101 Profibus interface card

MCA101 (**Figure 46**) is a gateway that provides Profibus DP V1 interface for the drive. It is physically installed in the front of the drive main body just behind the LCP102 keypad module. It is capable of manually setting Profibus address and termination via DIP switches. The major functions it provides are listed below:

- Support both DP V0 and DP V1;
- Full range of baud rate from 19.3kBaud to 12MBaud;
- Support FC and PROFIdrive profile;
- MCT 10 support via DP V1;
- Configure process data;
- Access and control all I/O terminals. [52]

The Danfoss drive also supports other fieldbus communications, the available gateways are listed in **Table 7**.

Fieldbus	Gateways
Profibus	MCA101
DeviceNet	MCA104
CAN Open	MCA105
LonWorks	MCA108
BACnet	MCA109

Table 7: Available gateways for various fieldbus systems [55]

(Refer to Appendix VIII HVAC FC102 Configuration Instructions for module and configuration information)

MCF104 Profibus Adaptor Sub-D9 Connector

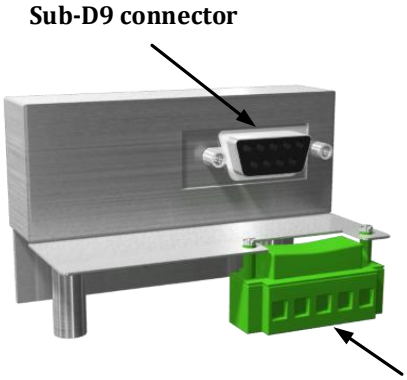


Figure 47: MCF104 Profibus adaptor Sub-D9 connector

The MCF104 module is actually a Sub-D9 adaptor compatible with the MCA101 gateway that converts the terminals of MCA101 into a standard Sub-D9 connector. With the help of MCF104, the Profibus connector can now be used instead of wires from the Profibus cable and bus termination can be set from the Profibus connector externally. [52]

(*Refer to Appendix VIII HVAC FC102 Configuration Instructions for module and configuration information*)

FC Protocol

The FC protocol is the standard fieldbus from Danfoss. It defines a data exchange technique based on the master-slave communication via the serial bus. One master can support up to 126 slaves on the bus and the master chooses the individual slave via an address character in the telegram. This single master system has a half-duplex transmission that does not allow data to be transferred between 2 slaves. [56]

Motion Control Tools 10 (MCT 10)

Figure 48: MCT 10 logo

MCT 10 (**Figure 48**) is a software package designed as an interactive tool for Danfoss drives. It has the capability to simultaneously control and configure the drive and effectively monitor multiple drives. MCT 10 also supports Profibus DP V1 as a class 2 master that allows it to access the drive parameters for easy operation and diagnosis. MCT 10 has the following highlighted features:

- Set up a network offline with its complete drive database;
- Online commissioning on multiple drives;
- Compatible with multiple protocols;
- Backup parameter data;
- Clear interface with configurable views;
- Efficient drive diagnosis. [50]

(Refer to Appendix VIII HVAC FC102 Configuration Instructions for MCT 10 configuration)

Slave Modules

Figure 49 shows all the slave modules in TIA Portal and the structure of the available slave modules is illustrated in **Figure 50**.

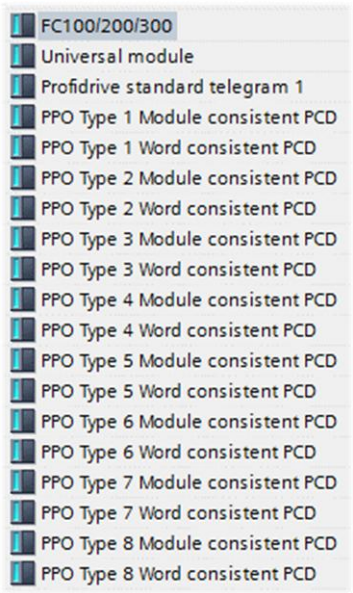


Figure 49: Slave modules of HVAC FC102 in TIA Portal

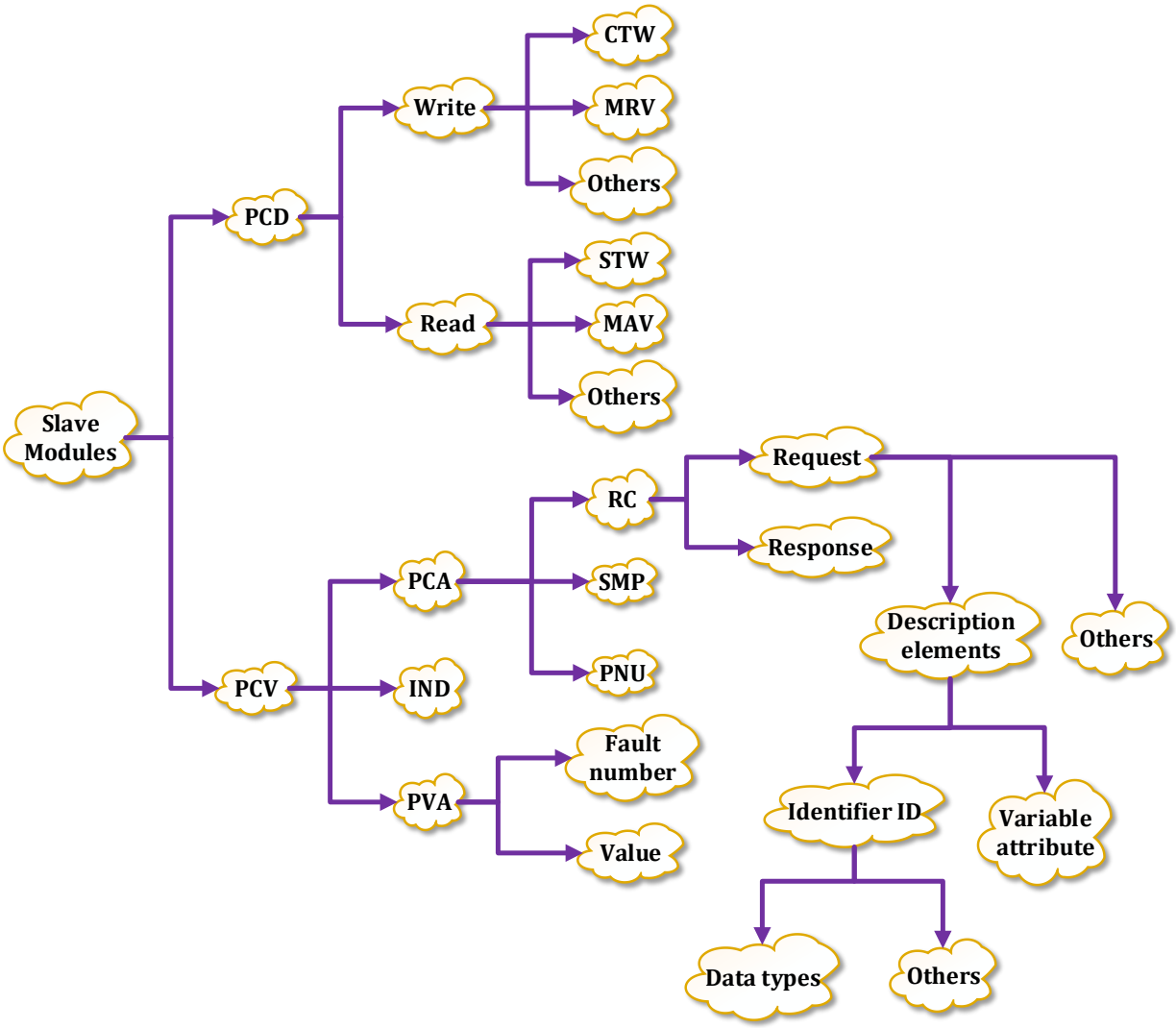


Figure 50: Overall structure of HVAC FC102 slave modules

PROFIdrive Standard Telegram 1 (Cyclical)

The PROFIdrive standard telegram works under the PROFIdrive profile via Profibus DP V1 communication. It is a standard type 1 PROFIdrive slave module for speed control that consists of a total length of 2 words for both input and output. The operation of the drive is controlled via the control word and its status is displayed in the status word. The operation state of the drive follows the PROFIdrive state diagram that will be explained in the next section. [12]

The PROFIdrive state diagram illustrates a generic series of sequential functions that a drive follows. It does not contain all the functions that a process word should have, but it provides a universal standard for the basic operations. The Danfoss HVAC drive is capable of performing these operations when choosing PROFIdrive profile for Profibus communication. **Figure 51** demonstrates this state diagram. [53]

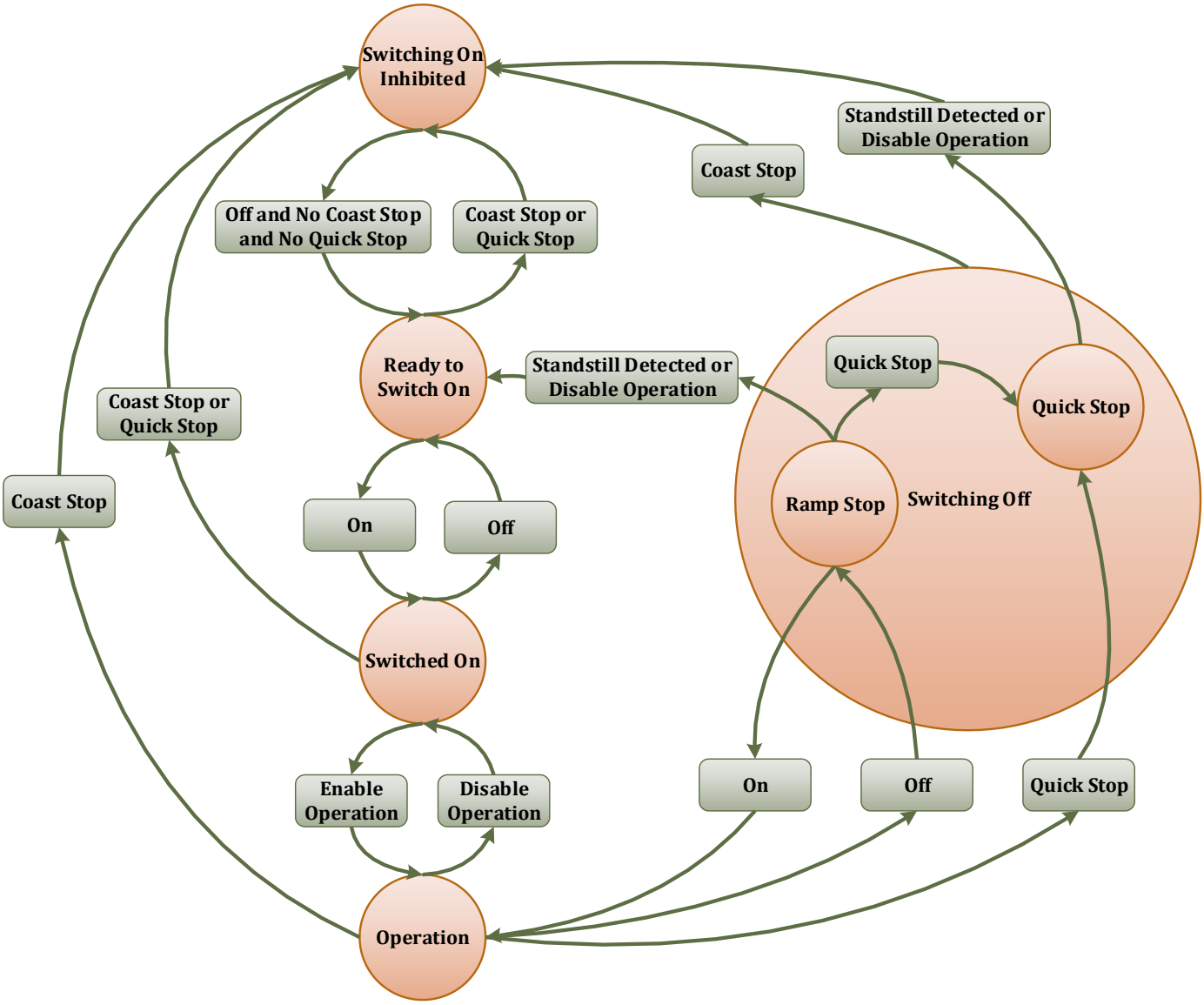


Figure 51: PROFIdrive state diagram [10] [53]

PPO Types (Cyclical)

A total number of 8 PPO (Parameter Process data Objects) type modules are available with different configurations and data lengths. PPO type 1, 2 and 5 includes both PCV (Parameter Characteristics Value) and PCD (Process Control Data) channels and the remaining PPO type modules only contains PCD channel.

For each PPO type, there are two formats which are **Module consistent** and **Word consistent**. **Module consistent** means that a portion of the PPO is connected to the module and all the data need to be transferred at one time as a whole, whereas **Word consistent** allows an individual PCD in the PPO to be transferred at each time. [56]

(Refer to Appendix VIII HVAC FC102 Configuration Instructions for Profibus data exchange configuration)

PROFIdrive Parameter Channel

DRVDPS7 library for STEP 7 contains a function block called **FB PDAT_AC2 (FB36)**: **Process parameters acyclically DS47**, which is responsible for the acyclic communication between the drive and the PLC for the parameter data transmission in accordance with PROFIdrive profile. This function block has a number of inputs and outputs which contains a DB for sending telegrams and a DB for reading telegrams. With the help of this PROFIdrive parameter channel, the master has read/write access to the value, attribute, and data format of multiple parameters in one transaction. However, this function block is not available in the current version of TIA Portal and no libraries containing this function block could be found. [9] [11] [16] [54]

DP/PA Coupler

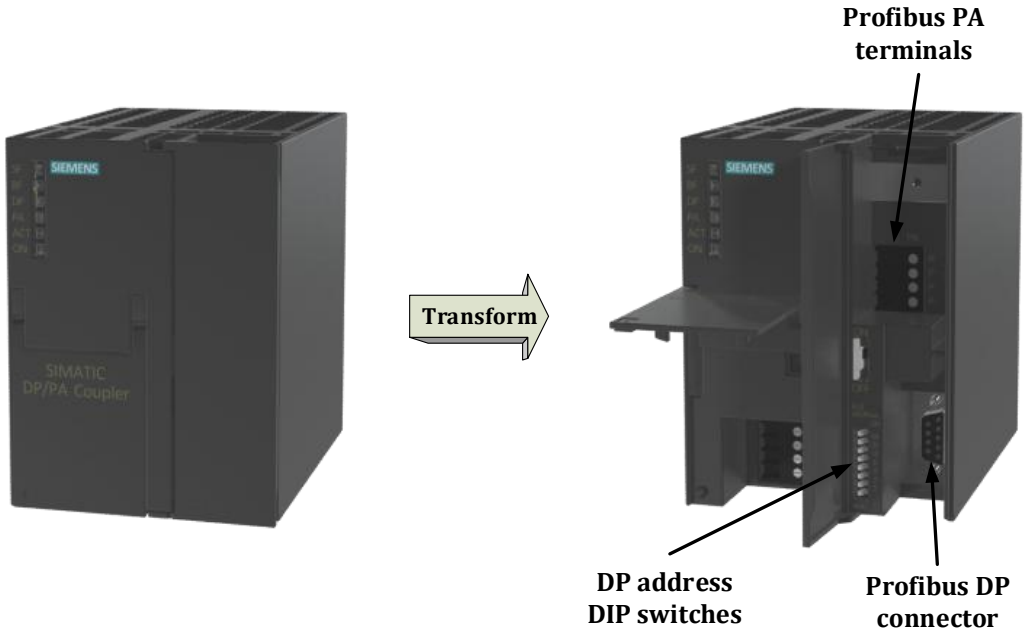


Figure 52: DP/PA coupler structure

The DP/PA coupler (**Figure 52**) is a physical link between Profibus DP and Profibus PA. It is able to address PA field devices via Profibus DP. The DP/PA coupler is mainly used for small volumes of data, whereas DP/PA link is suitable to handle large volumes of data. [80] Some highlighted feature of DP/PA are listed below:

- Communicate with Profibus DP with a transmission rate of 45.45 kBaud;
- Communicate with Profibus PA with a transmission rate of 31.25 kBaud;
- Supply power to Profibus PA field device;
- Bus termination for Profibus PA;
- Support up to 31 PA devices. [32]

(Refer to Appendix XII PA System Configuration Instructions for module and configuration information)

Slave Modules

The available slave modules of DP/PA coupler are shown in **Figure 53**.

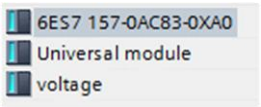


Figure 53: Slave modules of DP/PA coupler in TIA Portal

Current (Cyclical)

The **current** slave module is mandatory and shows the current on the equipotential bonding line. [80]

(Refer to Appendix XII PA System Configuration Instructions for Profibus data exchange configuration)

Voltage (Cyclical)

The **voltage** slave module displays the voltage value on the equipotential bonding line. This module is not deliverable on the DP/PA coupler standalone mode. [80]

Levelflex M FMP40

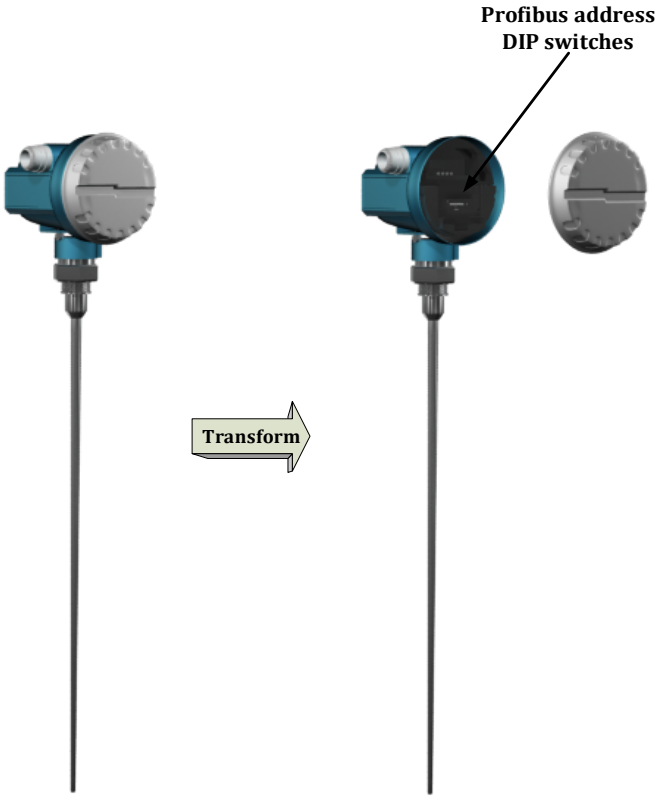


Figure 54: Levelflex M FMP40 structure

The available Endress+Hauser level transmitter model is Levelflex M FMP40 (**Figure 54**). This level transmitter has a Profibus PA interface and is capable of continuous liquid and solid level measurements. The physical set up for liquid measurement is shown in **Figure 55**.

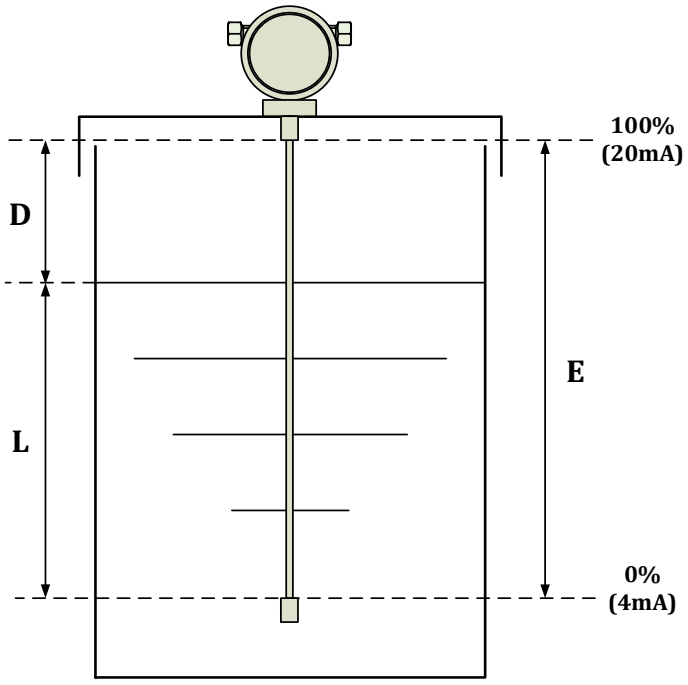


Figure 55: Level measurement set up diagram [82]

(Refer to Appendix XII PA System Configuration Instructions for module and configuration information)

Measuring Principle

The Levelflex M FMP40 is a measuring system that operates based on the time of flight (TOF) principle. The distance between the reference point and the product surface is measured. High frequency pulses are transmitted to the measuring probe and the pulses are then reflected by the product surface. The microprocessor in the transmitter then evaluates and converts the time of flight into the level information. [82]

Based on **Figure 55**, the equation to calculate the level is given by:

$$L = E - \frac{c \times t}{2}$$

Equation 1: Levelflex M FMP40 level measurement calculation [82]

Where L = Actual level

E = Designed lowest level

c = Speed of light

t = Time of flight [82]

Slave Modules

Figure 56 shows all the slave modules in TIA Portal.

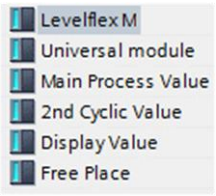


Figure 56: Slave modules of Levelflex M FMP40 in TIA Portal

Main Process Value (Cyclical)

Display the scaled actual value of the measurement. [81]

2nd Cyclic Value (Cyclical)

Display the distance between the sensor membrane and the measuring surface. [81]

Display Value (Cyclical)

Value sends from PLC to the transmitter to be displayed on the on-site LCD module. [81]

Free Place (Cyclical)

This module must be placed on the module rack if there is an empty slot. [81]

(Refer to Appendix XII PA System Configuration Instructions for Profibus data exchange configuration)

Deltabar S PMD70

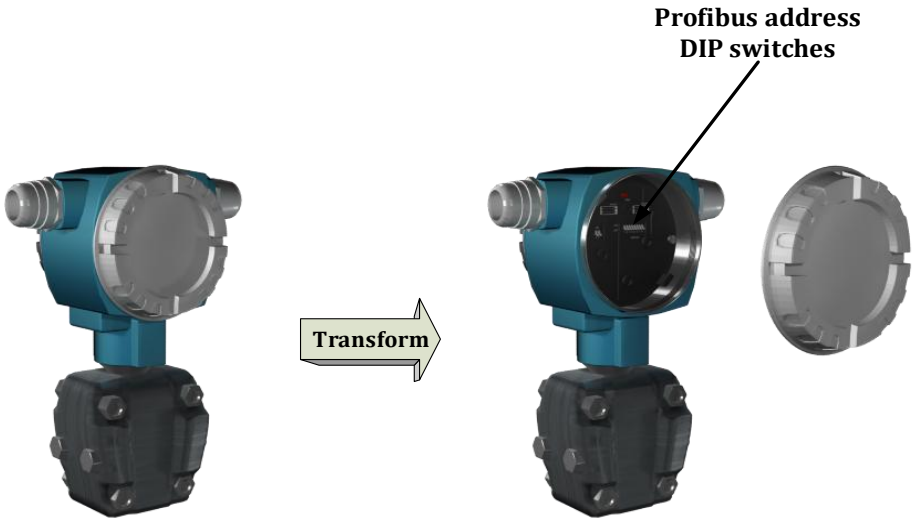


Figure 57: Deltabar S PMD70 structure

The Deltabar S PMD70 pressure transmitter has a measuring range of -25mbar to +25mbar and is suitable for the following applications:

- Volume/mass flow measurements;
- Volume, level and mass measurements in liquid;
- Differential pressure monitoring;
- Ambient temperature measurement. [19] [84]

The physical set up for simple level measurement on an open tank is shown in **Figure 58**.

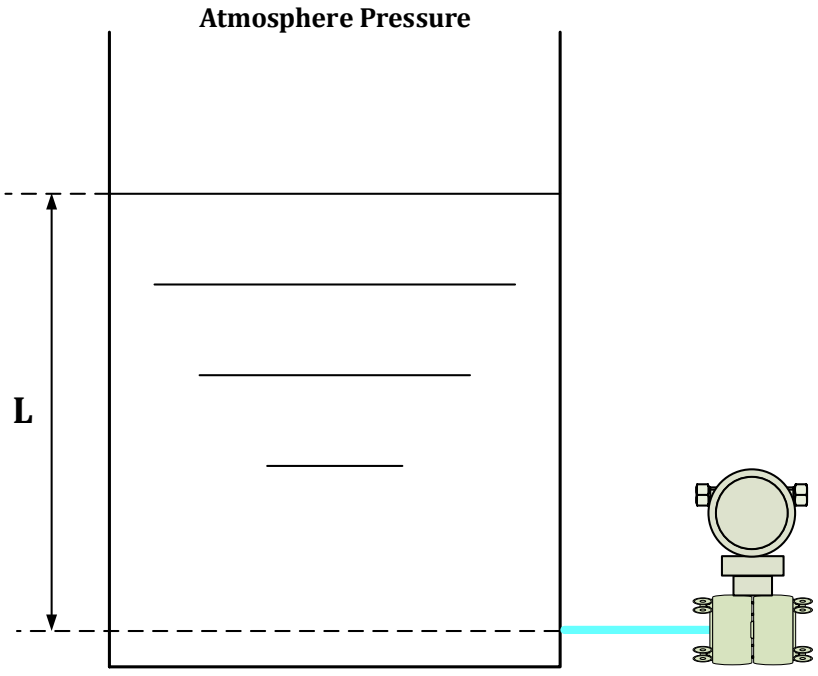


Figure 58: Level measurement set up diagram [84]

(Refer to Appendix XII PA System Configuration Instructions for module and configuration information)

Measuring Principle

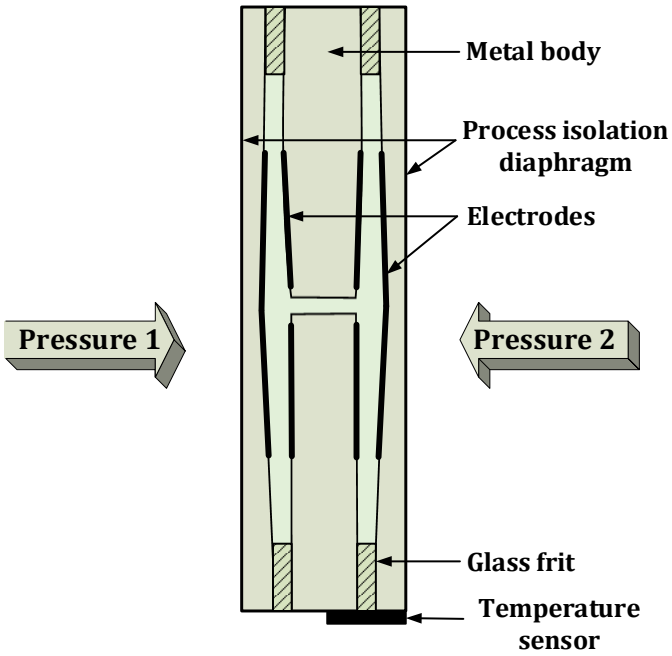


Figure 59: Measuring principle diagram [84]

The principle of the measuring method is a plate capacitor with an electrode on the meter body and an electrode on the interior of the diaphragm which is movable. Whenever there is a differential pressure, it causes a deflection of both diaphragms and both capacitance values are converted and fed to the microprocessor as shown in **Figure 59**. [84]

Slave Modules

Figure 60 shows all the slave modules in TIA Portal.

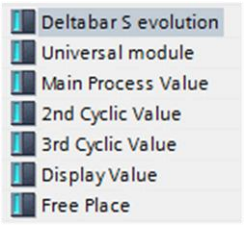


Figure 60: Slave modules of Deltabar S PMD70 in TIA Portal

Main Process Value (Cyclical)

This slave module displays the scaled actual value of the measurement. Three types of measurements could be chosen in either the on-site LCD module or FieldCare: pressure, level and flow. [83]

2nd Cyclic Value (Cyclical)

This slave module display the value of the measurement depends on the option selected in either the on-site LCD module or FieldCare. Four available options are as follows:

- **Temperature**: Corresponds to the temperature parameter (factory setting);
- **Sensor value**: Corresponds to the sensor pressure parameter;
- **Trimmed value**: Corresponds to the corrected pressure parameter;
- **Secondary value**: Corresponds to the pressure parameter. [83]

3rd Cyclic Value (Cyclical)

This displays the value of the measurement depends on the option selected in either the on-site LCD module or FieldCare. Two available options are as follows:

- Totalizer 1 (factory setting);
- Totalizer 2. [83]

Display Value (Cyclical)

This value is sent from PLC to the transmitter to be displayed on the on-site LCD module. [83]

Free Place

This module must be placed on the module rack if there is an empty slot. [83]

(Refer to Appendix XII PA System Configuration Instructions for Profibus data exchange configuration)

RS485 Repeater

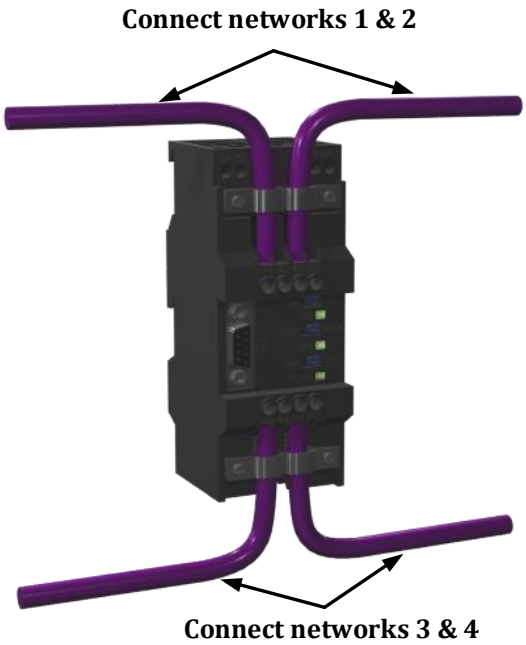


Figure 61: RS485 repeater structure

The repeater (**Figure 61**) is typically used to connect 2 bus segments and amplifies the data signals to improve amplitude and edge slope on the bus line. A network must use a repeater if 32 nodes are connected to the bus or the maximum cable length limit is exceeded. Within a single network, up to 9 repeaters can be connected in series if no other nodes are installed between 2 repeaters. [40]

Figure 62 is the comparison of the Profibus signal received from a slave over a 10 meter Profibus cable with and without the repeater respectively.

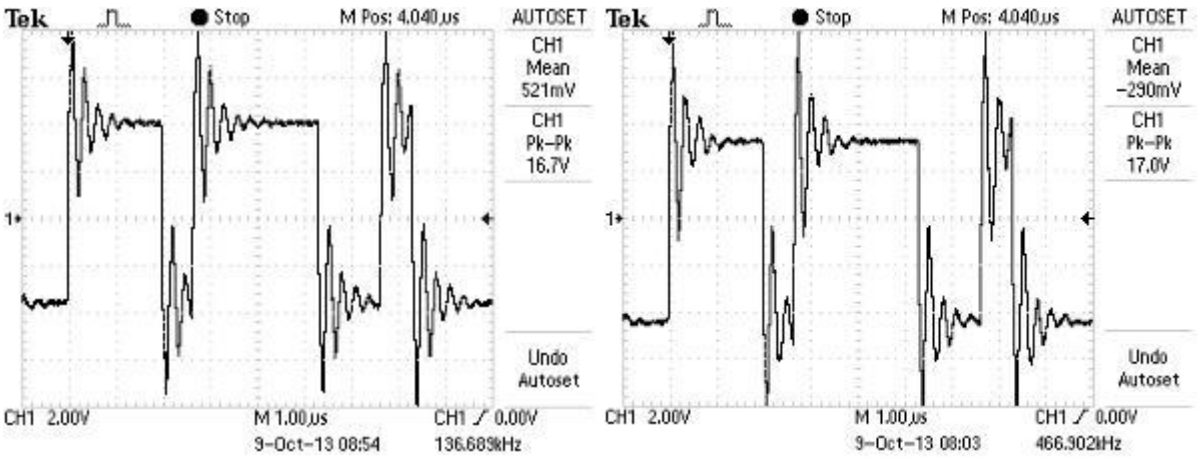


Figure 62: Comparison of the signal of a 10 meters Profibus cable with (left) and without (right) repeater

Table 8 shows the maximum distance with and without a repeater for different transmission speeds.

Transmission speed	Distance without repeater	Maximum distance with repeater
9.6 – 93.75 kbps	1000m	10km
187.5 kbps	800m	8km
500 kbps	400m	4km
1.5 Mbps	200m	2km
3 – 12 Mbps	100m	1km

Table 8: Maximum distance with and without repeater for different transmission speeds [40]

(Refer to Appendix V S7-300 PLC and Repeater Configuration Instructions for module and configuration information)

Project Panel

A portable project panel was set up for easy testing, configuration and operation as shown in **Figure 63**. The project panel includes a 240V to 24V power supply, 2 PLCs, 1 DP/PA coupler, 1 repeater, 1 HMI touch panel, 1 EASY719 relay controller with its gateway, a few LEDs, 1 Danfoss three phase VSD with its motor (behind the project panel), 2 SEW single phase VSDs with their common gateway and motors.

Figure 63: Portable project panel for testing and device configurations

CHAPTER 7: CONTROL SYSTEM DESIGN

Designed Control System

The control system designed and implemented consists of 3 SISO (Single Input Single Output) control loops, a level loop, a temperature loop and a pressure loop. **Figure 64** illustrates the process with a P & ID diagram.

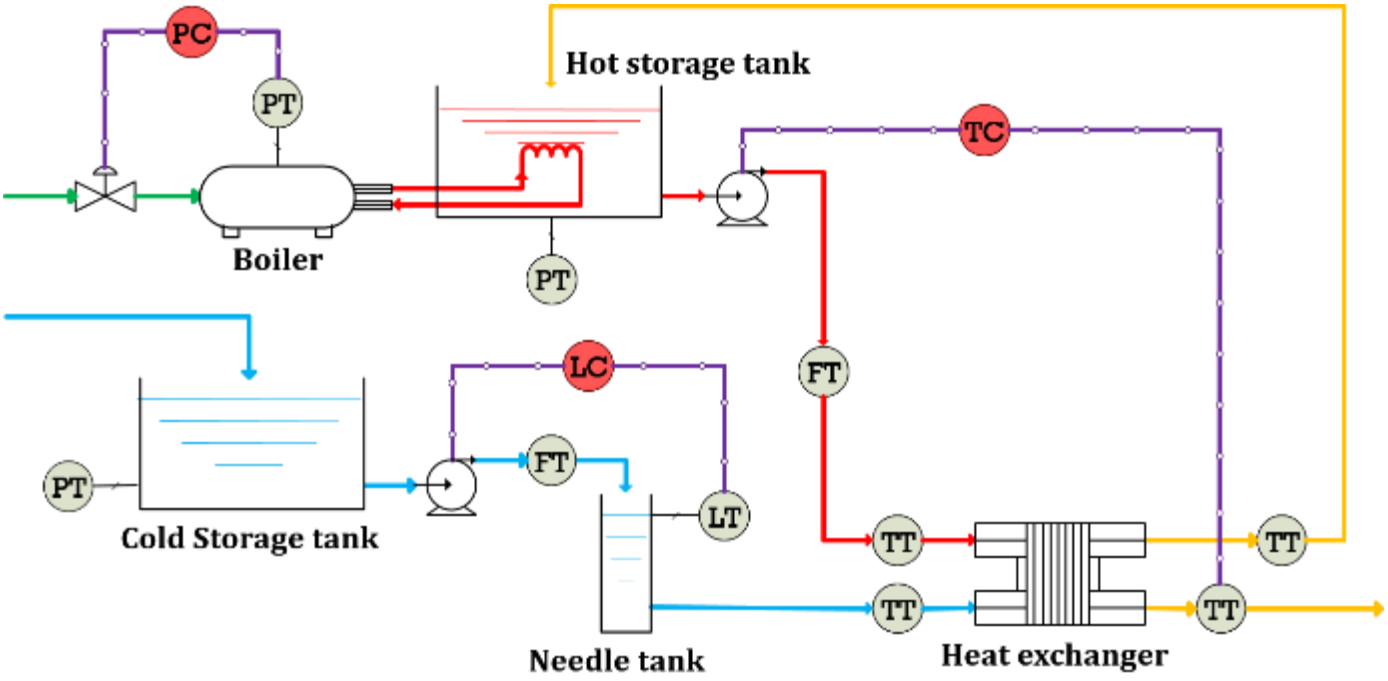


Figure 64: P & ID diagram of the designed control system

Level Control Loop

The cold storage tank holds a large volume of cold water and continuously pumps water into the needle tank. The water in the needle tank flows into the heat exchanger using gravity and the level of the needle tank is controlled by a cold water pump.

Temperature Control Loop

The temperature of the hot storage tank is supported by the steam from the boiler and the hot water is pumped into another heater exchanger input to transfer heat to the cold water stream. The temperature of the outlet cold water from the heat exchanger is the controlled variable and the control element for this temperature control loop is the hot water pump.

Pressure Control Loop

The pressure of the boiler is primarily controlled by the input control valve. The boiler continuously pumps steam into the hot storage tank and the steam flow rate depends on the boiler pressure. In short, the pressure of the boiler needs to be controlled by the input control valve and the boiler pressure has the impact on the water temperature in the hot storage tank.

Actual Available System

Figure 65: Actual system set up in the ICE laboratory

Since not all the instruments in the designed process are available in the laboratory, the actual system was set up in a very similar way to simulate the designed process (**Figure 65**). Below is the summary of the differences between the designed system and the actual system:

- The control valves were used as control elements instead of pumps;
- The pressure tank was standalone and did not have steam involved. Thus, it would not affect the water temperature in the hot storage tank;
- The water in the hot storage tank was heated by the internal tank heater coil not the steam.

(Refer to Background section in the thesis report for an overview of the ICE laboratory)

Network Overall Structure

Figure 66 shows the whole network structure of the actual control system.

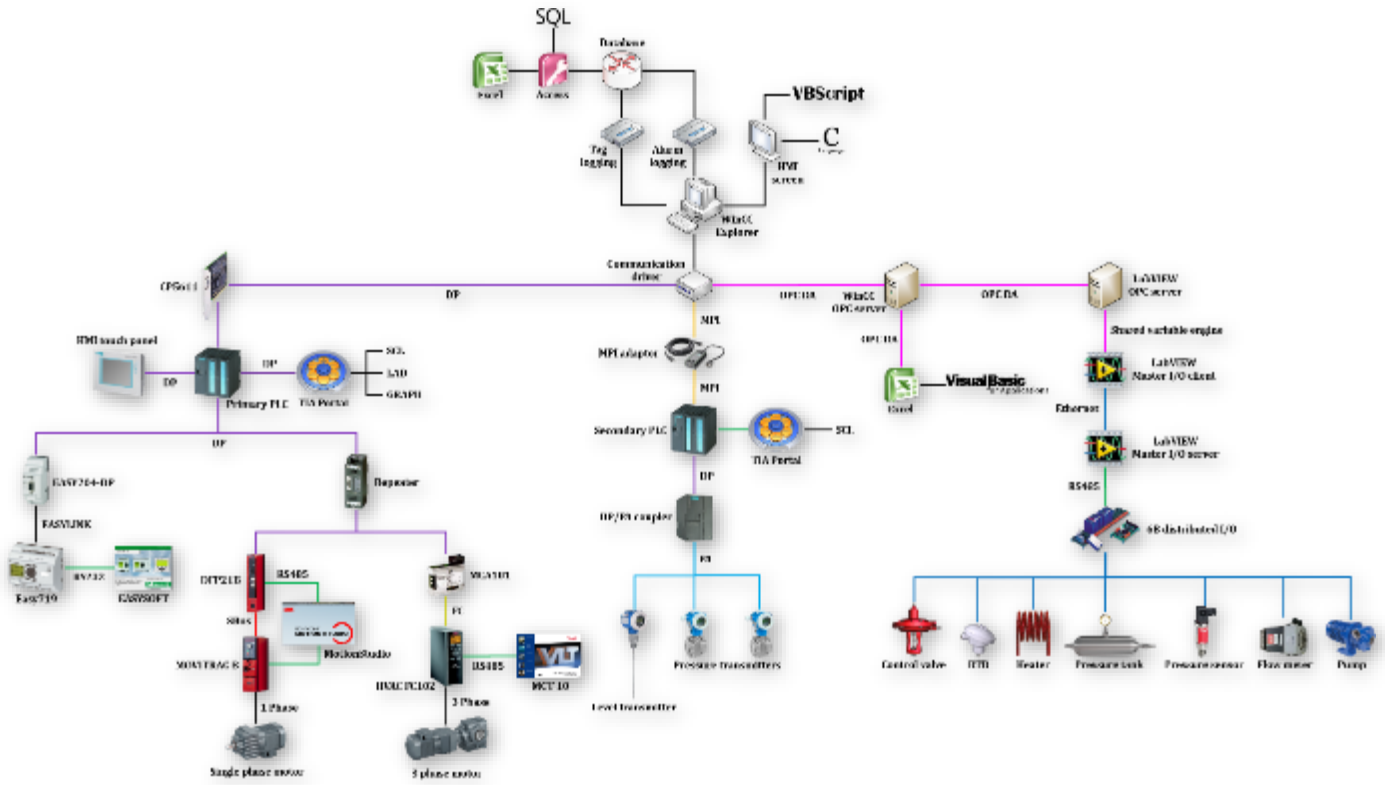


Figure 66: Overview of the whole network structure of the actual control system

The primary PLC communicated with WinCC via Profibus using a CP5611 Profibus communication processor. Under the primary PLC, a gateway was used to provide the Profibus interface for an EASY719 controller and a repeater was integrated after that to amplify the fieldbus signal. The output signal from the repeater was then connected to both SEW and Danfoss drives and each of the drive was connected to a motor.

The secondary PLC was connected to WinCC via an MPI network using a USB MPI adaptor. The PLC was then connected to a DP/PA coupler via Profibus DP communication and collected the data from level and pressure transmitters over Profibus PA protocol.

WinCC accessed LabVIEW shared variables from an OPC server. The Master I/O server LabVIEW program communicated with 6B distributed I/O modules to controlled the instruments in the laboratory.

WinCC monitored and controlled the whole process via HMI screens. The process values and alarm messages were also continuously logged into the WinCC database.

Detailed Description of Network Elements

The whole network structure can be divided into several parts and the detailed functionality of individual devices will be explained in the following sections.

Communication Driver

Three communication drivers used in WinCC were **SIMATIC S7 Protocol Suite**, **OPC** and **System**. The **SIMATIC S7 Protocol Suite** communication driver was used to provide Profibus and MPI communication. The **OPC** communication driver was apparently used to communicate with LabVIEW shared variable engine. At last, the **System** communication driver supported information about the computer RAM usage, drive free space, etc.

(Refer to Appendix XIII WinCC Configuration Instructions for communication driver setup)

Primary System

Figure 67 is the network structure of the primary system as part of the whole system.

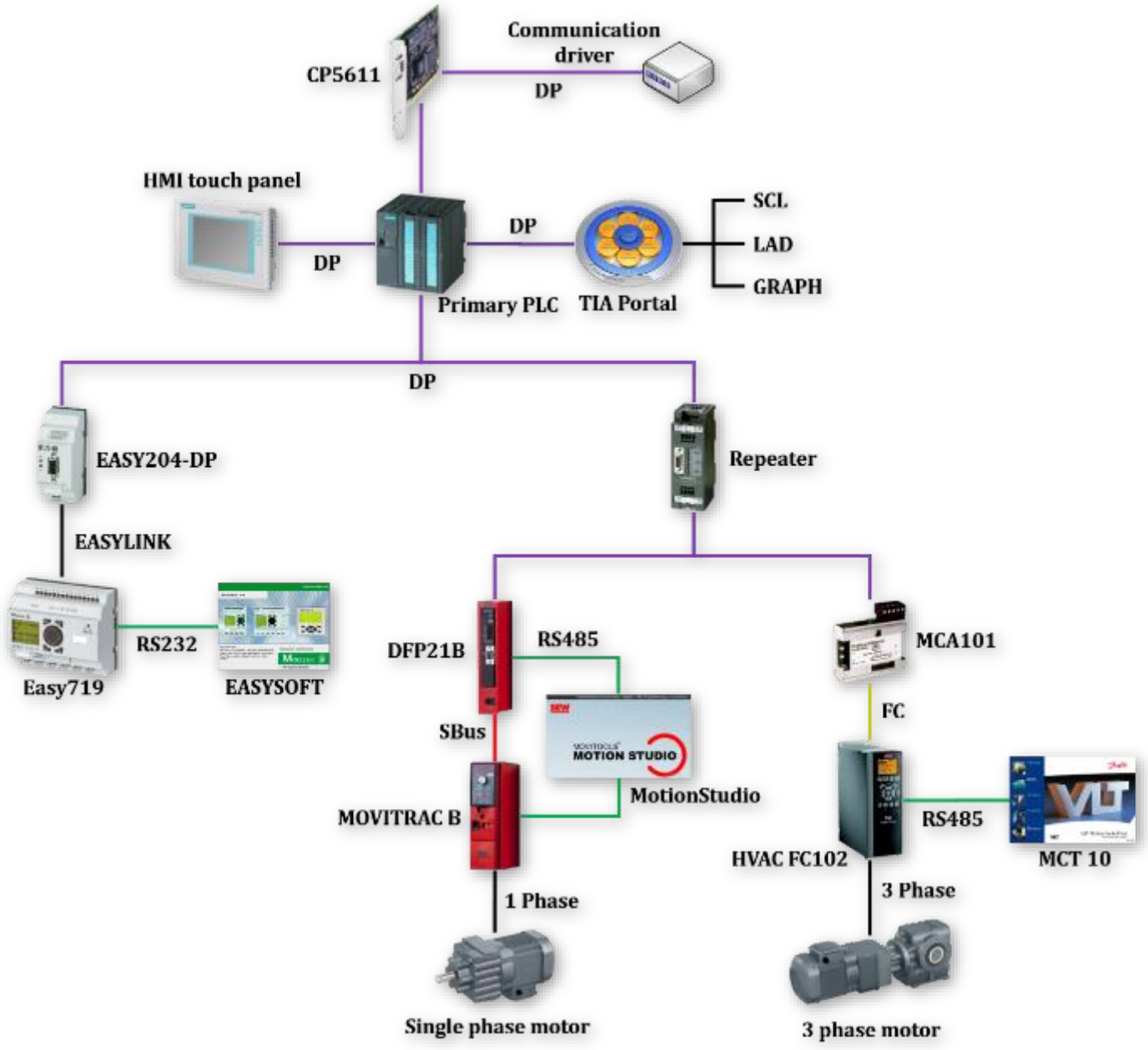


Figure 67: Primary system network structure

CP5611 Profibus Communication Processor

The CP5611 Profibus communication processor provided a communication path between WinCC and the primary PLC and also supported the PLC configuration in TIA Portal.

Primary PLC

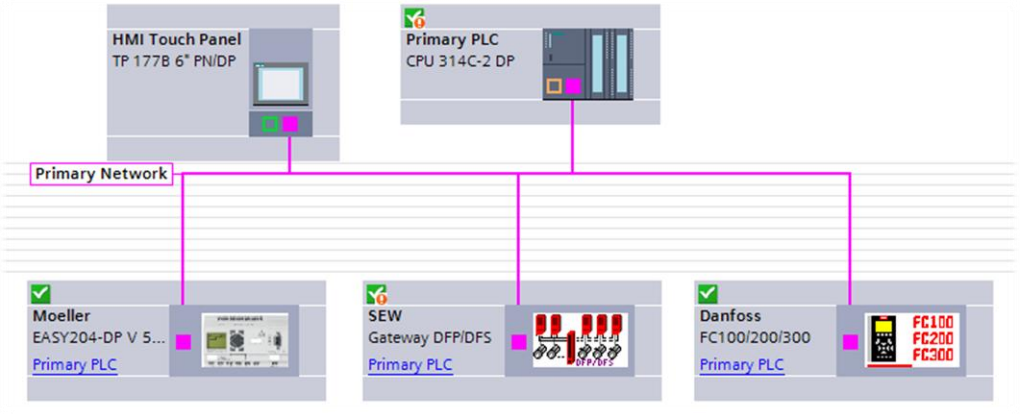


Figure 68: Primary system online network view in TIA Portal

The primary PLC was a master in the primary network instigated the communication with TP 177B 6" HMI touch panel, EASY719 controller, SEW and Danfoss drives (**Figure 68**). It also behaved as a central control center to manipulate the process data. All the calculations, conversions and commands of its slaves were sorted here and stored in the data blocks for WinCC to access.

TIA Portal was used as the configuration software for PLC and network devices. Due to the complexity of the data exchange, which involved a lot of data type conversions, bit shifting and masking, the PLC program was primarily written in the SCL text based language with a small section of LAD and GRAPH. **Figure 69** shows the SCL code to build PCA word for Danfoss parameter characteristics.

```
// Build PCA
#PCA := INT_TO_WORD(#ParameterNumber);
CASE #Request OF
  1: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[4],N := 1); // 0001
  2: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[5],N := 1); // 0010
  3: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[4],N := 2); // 0011
  4: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[6],N := 1); // 0100
  5: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[4],N := 1); SET(S_BIT := #RequestBits[6],N := 1); // 0101
  6: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[5],N := 2); // 0110
  7: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[4],N := 3); // 0111
  8: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[7],N := 1); // 1000
  9: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[4],N := 1); SET(S_BIT := #RequestBits[7],N := 1); // 1001
  ELSE RESET(S_BIT := #RequestBits[3],N := 5);
END_CASE;
```

Figure 69: Build PCA word for Danfoss PCV using SCL in TIA Portal

The entire PLC program was written in OB35, a constant cyclical executed organization block, to obtain a consistent constant execution rate of 250ms with HMI touch panel, WinCC and LabVIEW. Each slave device had a FB (Function Block) that took all the input data from the device and sent all the output data to the device. **Table 9** summaries the contents of all the program blocks in the primary PLC.

Block	Description
Moeller FB	Conversions and calculations of EASY719 controller I/O.
SEW FB	Conversions and calculations of SEW drive PD data and gateway status.
Danfoss FB	Conversions and calculations of Danfoss drive PCD data and PCV parameter value exchange.
System FB	Build warning word and error word and store status information for all the devices.
Toggle FB	PLC heart beat written in LAD and GRAPH.
PID DB	Stores process value and PID tuning parameters.

Table 9: Description of primary PLC program blocks in TIA Portal

(Refer to Appendix III Program Code for the FB program in the Primary PLC)

HMI Touch Panel

The TP 177B 6" HMI touch panel was used as a remote operator controller and was bound to the primary PLC. The HMI touch panel was designed to monitor the general operations of the process and change process setpoints and tuning parameters.

Screen

Figure 70 shows the overview and the control screen on the HMI touch panel. The overview screen displays values for the major process variables and the operator can touch a certain object on the HMI touch panel to navigate to the corresponding screen. The operator can also change the screens using either the drop down list on the top or the screen buttons at the bottom.

The control screen has entries for tunning parameters of 3 control loops, with toggle switches to switch between manual and auto control.

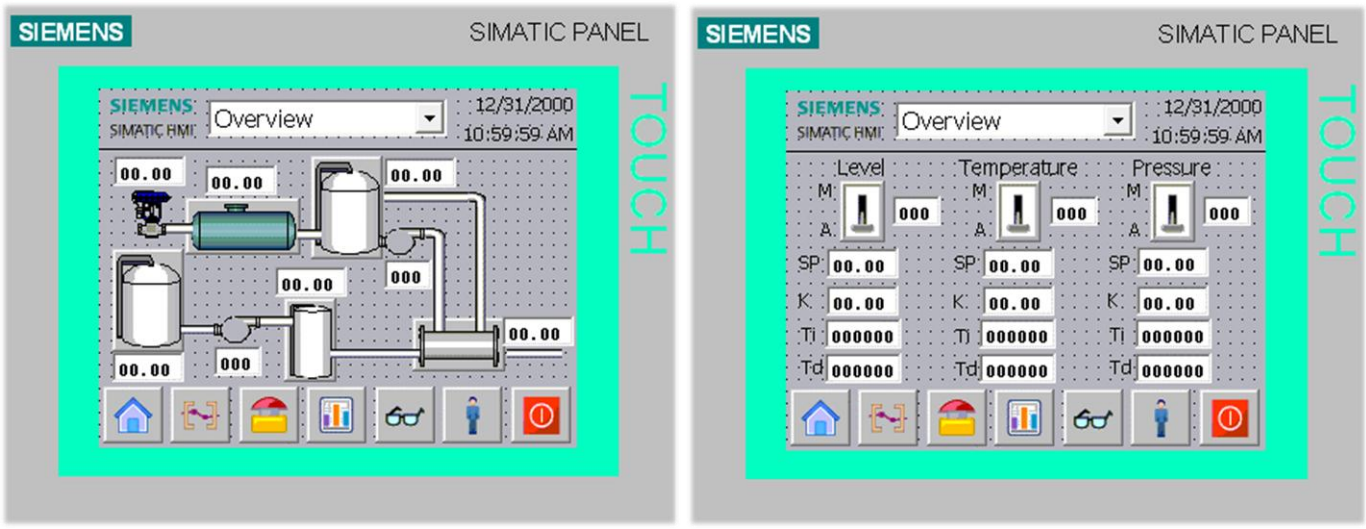


Figure 70: HMI touch panel Overview (left) and Control (right) screens

The alarm screen in **Figure 71** displays alarm messages if the conditions of certain alarms were triggered. The **Loop-in-Alarm** button was available to navigate to the screen which contains the variable that triggers the alarm. Operator can acknowledge the alarm by clicking the acknowledge button underneath.

The force/monitor screen provides access to every variable in the bound PLC with the highest priority. It was generally used when an emergency occurs or some special purposes.

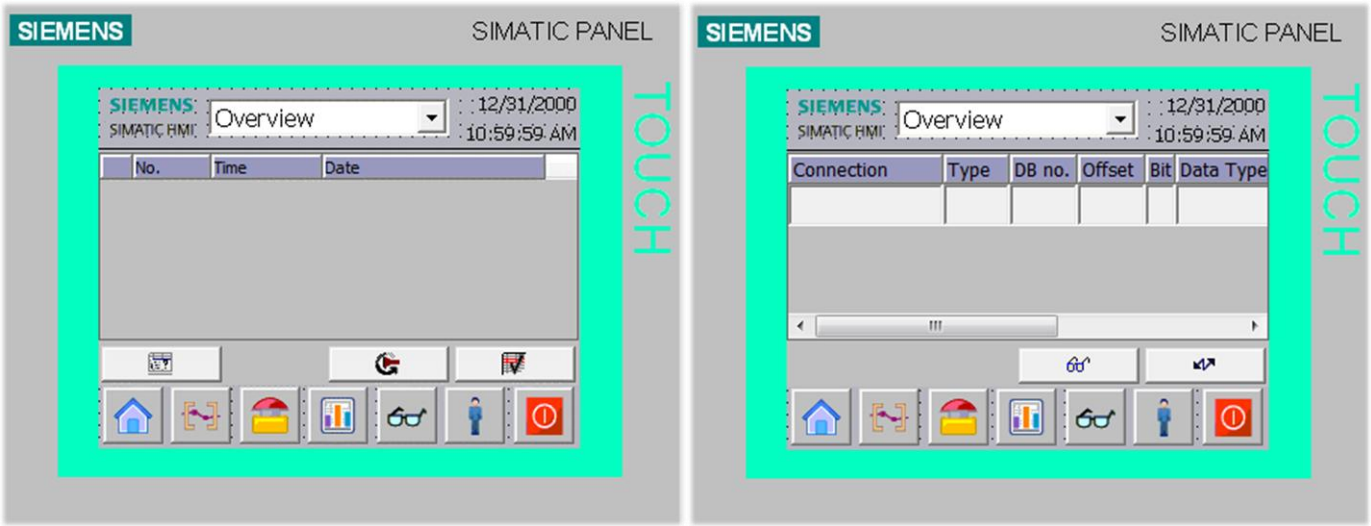


Figure 71: HMI touch panel Alarm and Force/Monitor screens

Figure 72 shows the screen for cold and hot storage tanks and the arrow button navigates to the previous or the next process screen.

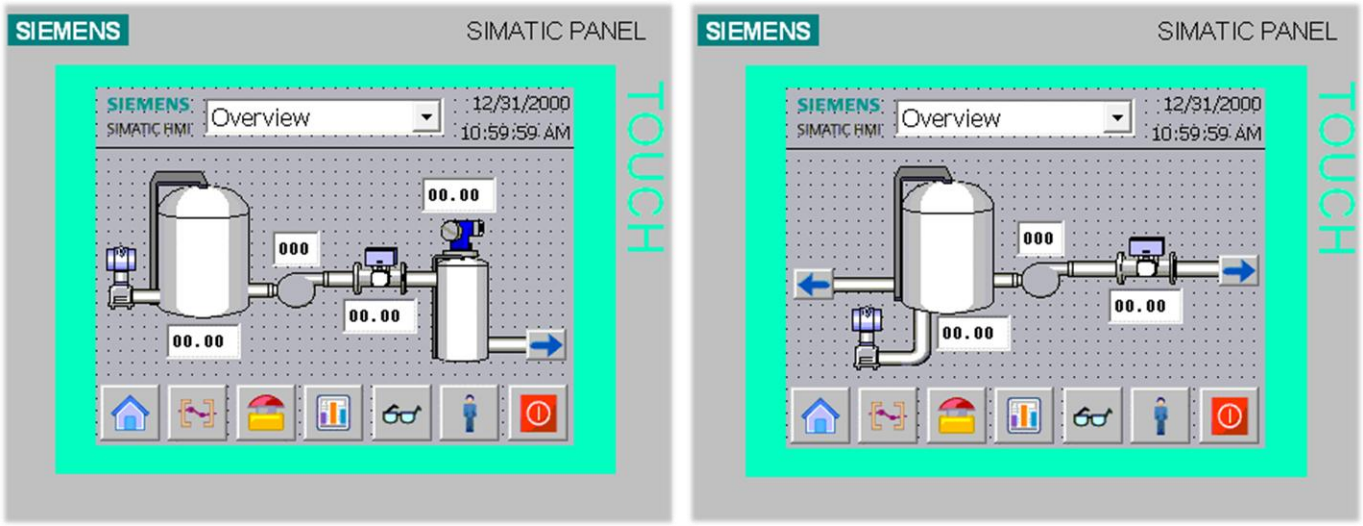


Figure 72: HMI touch panel Cold storage tank (left) and Hot storage tank (right) screens

The pressure tank and the heat exchanger screens are shown in **Figure 73**.

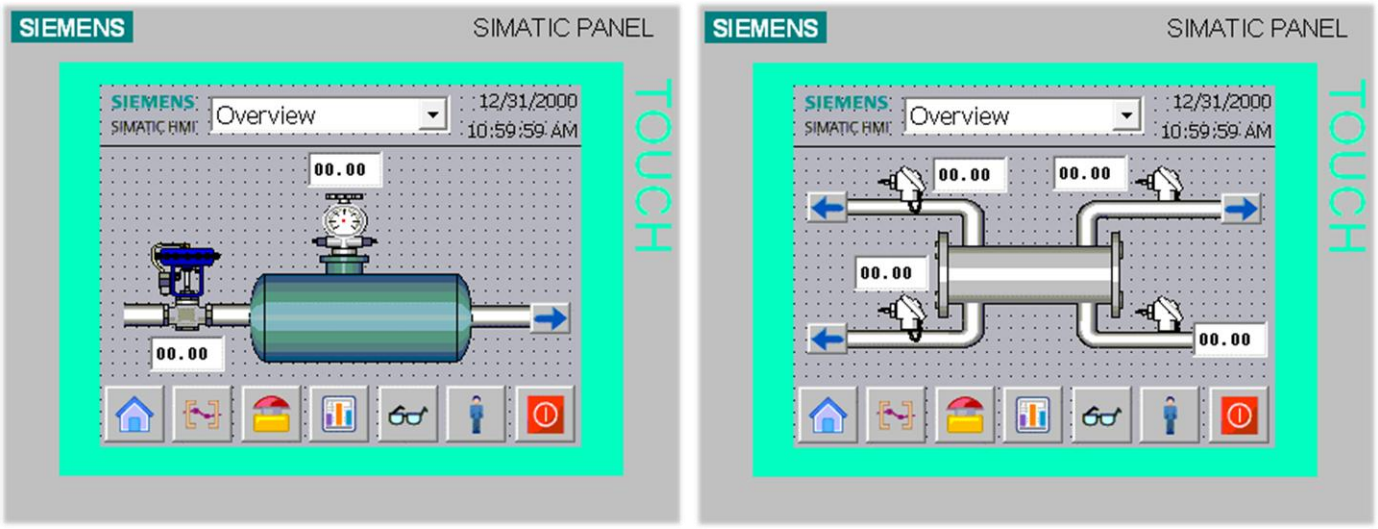


Figure 73: HMI touch panel Pressure tank (left) and Heat exchanger (right) screens

Due to the limitation of the physical size of the HMI touch panel, the trending for each control loop was configured to display on an individual screen as shown in **Figure 74**. When the **Trending** icon at the bottom is pressed, the user can choose the desired control loop to monitor.

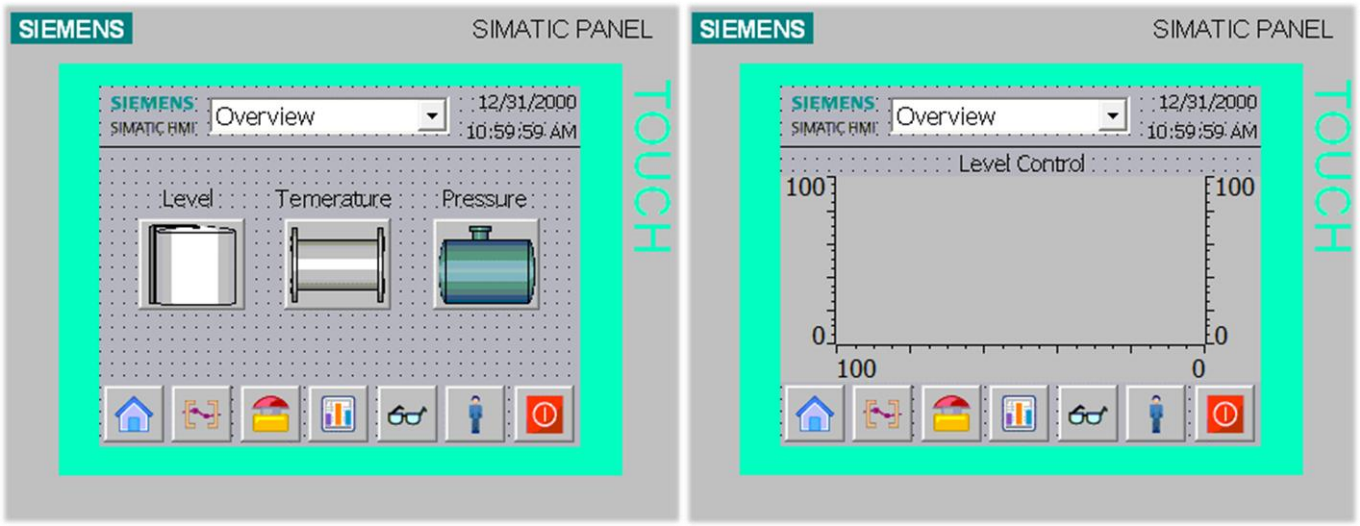


Figure 74: HMI touch panel Trend selection (left) and Level trending (right) screens

Alarm

Figure 75 shows all the alarms available in the HMI touch panel.

Discrete alarms

ID	Alarm text	Alarm class	Trigger tag	Trigger bit
1	Needle tank level low	Warnings	SystemDB_WarningWord	0
2	Needle tank level high	Warnings	SystemDB_WarningWord	1
3	Output temperature low	Warnings	SystemDB_WarningWord	2
4	Output temperature high	Warnings	SystemDB_WarningWord	3
5	Steam pressure low	Warnings	SystemDB_WarningWord	4
6	Steam pressure high	Warnings	SystemDB_WarningWord	5
7	Hot storage tank level low	Warnings	SystemDB_WarningWord	6
8	Hot storage tank level high	Warnings	SystemDB_WarningWord	7
9	Cold storage tank level low	Warnings	SystemDB_WarningWord	8
10	Cold storage tank level high	Warnings	SystemDB_WarningWord	9
11	SEW drive speed low	Warnings	SystemDB_WarningWord	10
12	SEW drive speed high	Warnings	SystemDB_WarningWord	11
13	Danfoss drive speed low	Warnings	SystemDB_WarningWord	12
14	Danfoss drive speed high	Warnings	SystemDB_WarningWord	13
15	PLC fault	Errors	SystemDB_ErrorWord	0
16	HMI fault	Errors	SystemDB_ErrorWord	1
17	Moeller fault	Errors	SystemDB_ErrorWord	2
18	SEW fault	Errors	SystemDB_ErrorWord	3
19	Danfoss fault	Errors	SystemDB_ErrorWord	4
20	DP/PA coupler fault	Errors	SystemDB_ErrorWord	5
21	Level transmitter level fault	Errors	SystemDB_ErrorWord	6
22	Level transmitter distance fault	Errors	SystemDB_ErrorWord	7
23	Hot pressure transmitter level fault	Errors	SystemDB_ErrorWord	8
24	Hot pressure transmitter temperature fault	Errors	SystemDB_ErrorWord	9
25	Cold pressure transmitter level fault	Errors	SystemDB_ErrorWord	10
26	Cold pressure transmitter temperature fault	Errors	SystemDB_ErrorWord	11

Figure 75: Discrete alarms in TP 177B 6" HMI touch panel

(Refer to Appendix X TP 177B 6" TIA Portal Configuration Instructions or Appendix XI TP 177B 6" WinCC flexible Configuration Instructions for configuration information)

EASY719 Controller & Gateway

The EASY719 controller was responsible for displaying 6 alarm conditions via its outputs which were connected to 6 external LEDs. The Profibus communication was through its own gateway EASY204-DP. **Table 10** shows the corresponding alarms.

Alarm	EASY719 output
Level high limit	Q1
Level low limit	Q2
Temperature high limit	Q3
Temperature low limit	Q4
Pressure high limit	Q5
Pressure low limit	Q6

Table 10: EASY719 alarms assignments

The controller was configured and programmed using EASYSOFT configuration software through RS232 communication. The simple program written is shown in **Figure 76**.

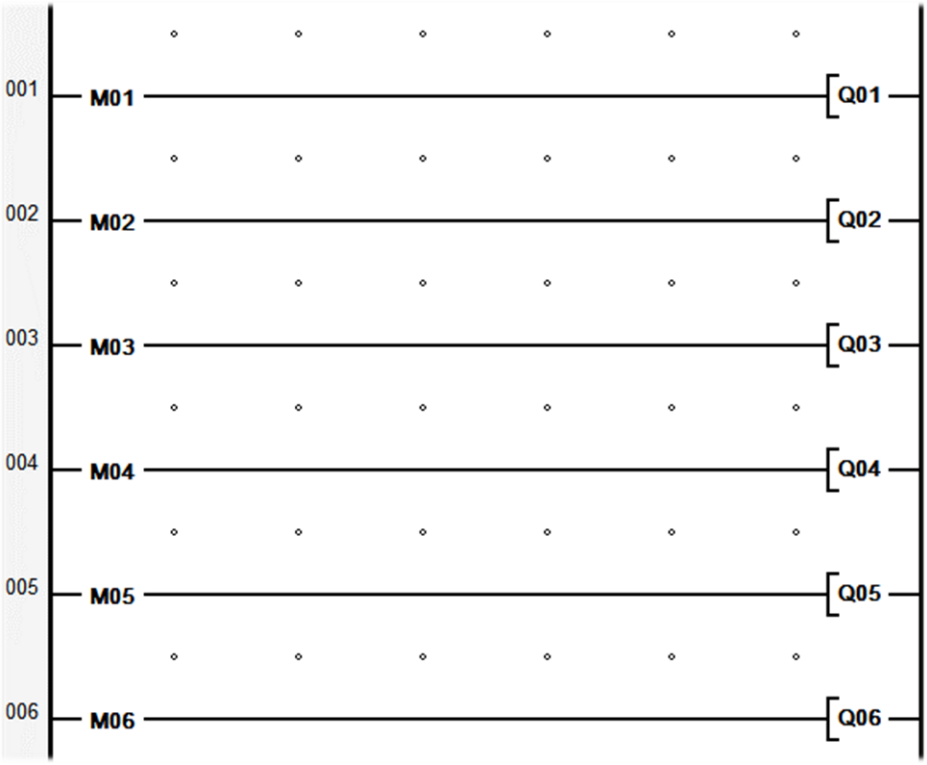


Figure 76: EASY719 program in EASYSOFT

Since EASY719 can only write to a single marker in each scan via the **EASY 700/800 control command** slave module, the longest delay time for an alarm display would be 6 times of the global scan cycle (250ms) which is 1.5 seconds.

(Refer to Appendix VI EASY719 & EASY204-DP Configuration Instructions for configuration information)

Repeater

The repeater was integrated into the network just after the EASY719 controller to boost the fieldbus signal for the following 2 drives.

(Refer to Appendix V S7-300 PLC & Repeater Configuration Instructions for configuration information)

SEW Drive and Gateway

The SEW drive Profibus data exchange was performed through a DFP21B gateway and the parameters were configured in MotionStudio configuration software. The job of the SEW drive and the connected motor were to simulate the response of the needle tank inlet pump in the designed control system and the actual output speed was scaled to a percentage value and also sent to the inlet control valve of the needle tank in the actual system. The assignments of **AS** slave module PI and PO data options are shown in **Table 11**.

Process data	Assignments
PI1	Status word 2
PI2	Actual speed
PI3	Actual speed [%]
PO1	Control word 2
PO2	Max. speed
PO3	Setpoint speed

Table 11: Process data assignments of MOVITRAC B

As **Table 11** indicates, control word 2 and status word 2 were selected to avoid the physical wire to enable the drive and also obtain more information about the drive and the motor from the virtual digital outputs. The virtual digital inputs and outputs assignments are listed in **Table 12**.

Virtual digital inputs	Assignments	Virtual digital outputs	Assignments
DI10	Enable/stop	**DO10**	Ready
DI11	CW/stop	**DO11**	Output stage ON
DI12	CCW/stop	**DO12**	Fault
DI13	n11/n21	**DO13**	Rotating field ON
DI14	n12/n22	**DO14**	Parameter set
DI15	Speed ramp switchover	**DO15**	Speed reference signal
DI16	Parameter set switchover	**DO16**	Setpoint-actual value comparison signal
DI17	Error reset	**DO17**	Imax signal

Table 12: Virtual digital inputs and outputs assignments of MOVITRAC B

(Refer to Appendix VII MOVITRAC B & DFP21B Configuration Instructions for configuration information)

Danfoss Drive

The Danfoss drive was the last slave in the primary Profibus network. Similar to the SEW drive, the Danfoss drive and the connected 3 phase motor were used to simulate the response of the hot storage tank output pump and the actual output speed was also sent to the outlet of the hot storage tank. The configuration of the drive was done in MCT 10 configuration software on a laptop for safety reasons. The assignments of FC protocol PPO type 5 slave modules are shown in **Table 13**.

PCD Write	Assignments	PCD Read	Assignments
PCD Write 1	Fieldbus CTW 1	**PCD Read 1**	Status word
PCD Write 2	Fieldbus REF 1	**PCD Read 2**	Main actual value [%]
PCD Write 3	Motor speed low limit [RPM]	**PCD Read 3**	Power [kW]
PCD Write 4	Motor speed high limit [RPM]	**PCD Read 4**	Motor voltage
PCD Write 5	Ramp 1 ramp up time	**PCD Read 5**	Frequency
PCD Write 6	Ramp 1 ramp down time	**PCD Read 6**	Speed [RPM]
PCD Write 7	Torque limit motor mode	**PCD Read 7**	Motor current
PCD Write 8	Digital & relay bus control	**PCD Read 8**	Heatsink temp.
PCD Write 9	Pulse out #27 bus control	**PCD Read 9**	Digital input
PCD Write 10	Terminal 42 output bus control	**PCD Read 10**	Digital output [bin]

Table 13: PCD assignments of HVAC FC102

(Refer to Appendix VIII HVAC FC102 Configuration Instructions for configuration information)

Secondary System

Figure 77 is the network structure of the secondary system as part of the whole system.

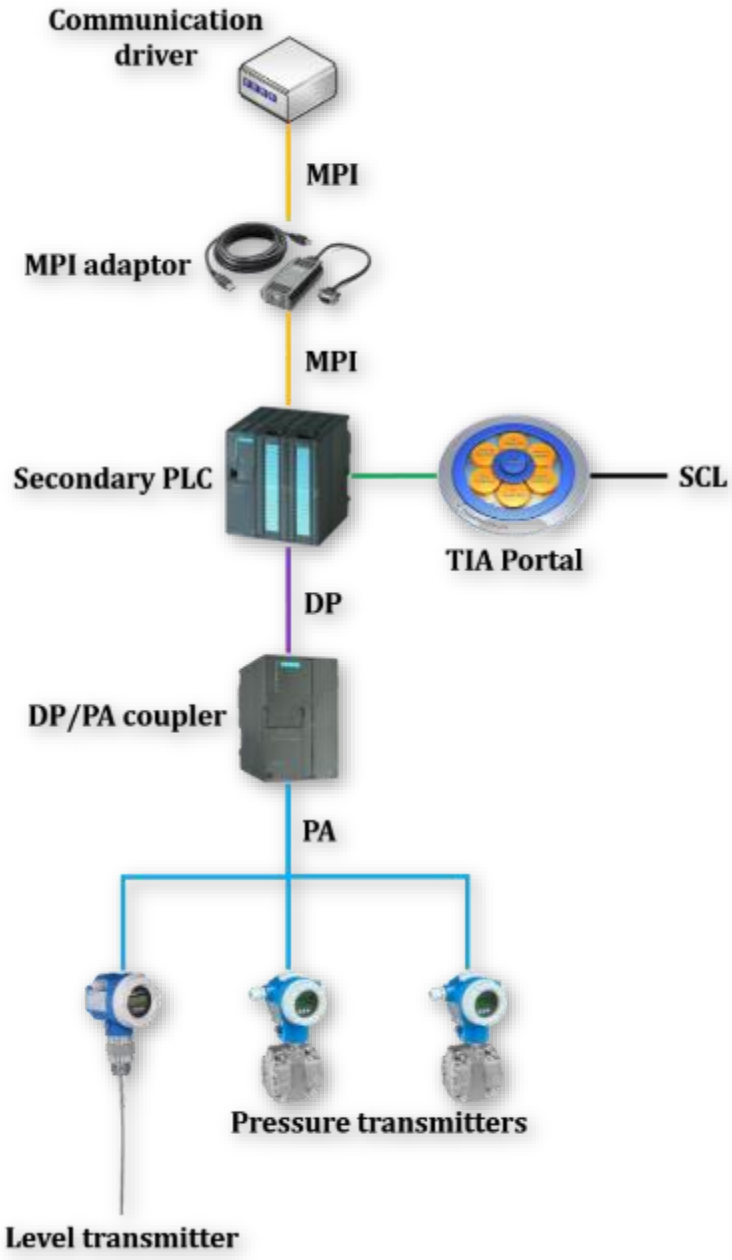


Figure 77: Secondary system network structure

MPI Adaptor

The MPI adaptor was used to establish MPI communication between the secondary PLC and the PC/WinCC. The program can be downloaded via MPI communication instead of Profibus communication to achieve a 4 fold increase in online communication speed (187.5kbps) due to the low DP/PA coupler transmission speed (45.45kbps).

Secondary PLC

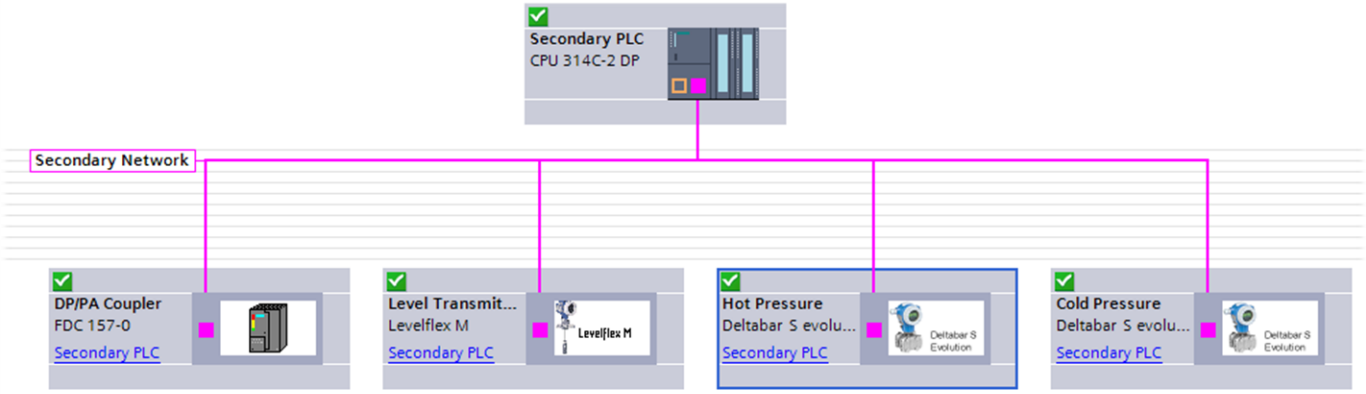


Figure 78: Secondary system online network view in TIA Portal

The secondary PLC was set up to collect the field transmitter data from the DP/PA coupler because the transmission speed (45.45kbps) of the DP/PA coupler on the Profibus side was not supported by other DP slaves. The secondary PLC was configured in TIA Portal with DP/PA coupler and field transmitters. Both PLCs were configured within the same project file in TIA Portal, but within different subnets.

The collected raw data for each slave were scaled and converted to a readable actual values using SCL within an individual function block.

Since there was only one CP5611 Profibus communication processor installed on the computer and TIA Portal does not support MPI global data communication, the data collected by the secondary PLC could not be transmitted to the primary PLC directly. Therefore, the MPI communication channel was used in WinCC to obtain those data and send the data to the primary PLC memory using ANSI-C actions.

Table 14 summaries the contents of all the program blocks in the secondary PLC.

Block	Description
DPPACoupler FB	Conversions and calculations of current and DP/PA coupler status.
LevelTransmitter FB	Conversions and calculations of level in needle tank and status.
PressureTransmitterHot FB	Conversions and calculations of level in hot storage tank and status.
PressureTransmitterCold FB	Conversions and calculations of level in cold storage tank and status.

Table 14: Description of secondary PLC program block in TIA Portal

(Refer to Appendix III Program Code for the FB program in the Secondary PLC)

DP/PA Coupler

The DP/PA coupler converted the PA data obtained from the field transmitters to DP data and stored them in the data block in the secondary PLC. The DP/PA coupler was connected to 1 level transmitter and 2 pressure transmitters with their M12 connectors plugged onto the I/O panels in the ICE laboratory. The DP/PA coupler equipotential current was read by the secondary PLC for condition monitoring purpose.

Level Transmitter

The level transmitter was physically inserted into the needle tank and continuously reading the tank level and the distance between the transmitter's base unit and the water surface.

Pressure Transmitter

There were 2 pressure transmitters used in the system to measure the level of the hot and cold storage respectively. They were physically connected to the bottom of the tanks via a tube. The pressure and the ambient temperature values were collected by the DP/PA coupler.

OPC System

Figure 79 shows the network structure of the OPC system as part of the whole system.

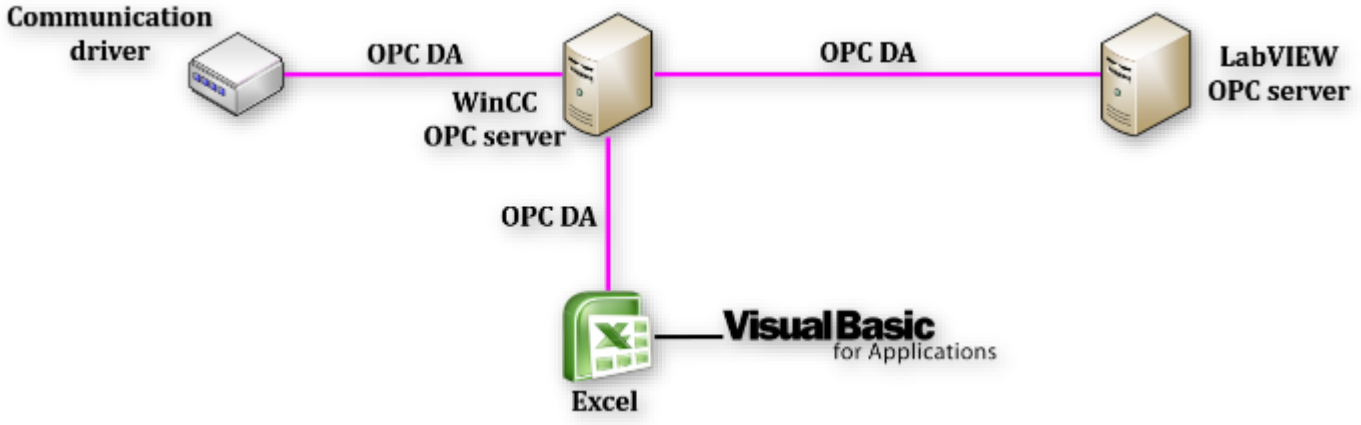


Figure 79: OPC system network structure

Excel WinCC Tag Monitoring

A large VBA program was developed using Siemens OPC DAAutomation2.0 add-on in Excel to access WinCC tag values. Most of the WinCC tags were manually entered in Excel and all the values were updated upon any changes made. The VBA program also allowed the user to overwrite values to WinCC tags directly when necessary. **Figure 80** is part of the VBA code to connect to the OPC server.

```
  ' Sub to connect to OPC server
Sub StartClient()
  ' Create ClientHandles
For i = 1 To 1
  ClientHandles(i) = i
Next i
  ' Create a group
  GroupName = "MyGroup"
  ' Define the computer name and tag names
  NodeName = Range("A1").Value
  ItemIDs(1) = Range("A2").Value
  ' Create a OPC instance
  Set MyOPCServer = New OPCServer
  MyOPCServer.Connect ServerName, NodeName
  Set MyOPCGroupColl = MyOPCServer.OPCGroups
  ' Set the default active state for adding groups
  MyOPCGroupColl.DefaultGroupIsActive = True
  ' Add group to the Collection
  Set MyOPCGroup = MyOPCGroupColl.Add(GroupName)
  Set MyOPCItemColl = MyOPCGroup.OPCItems
  ' Add items, return ServerHandles
  MyOPCItemColl.AddItems 1, ItemIDs(), ClientHandles(), ServerHandles(), Errors
  ' A group that is subscribed receives asynchronous notifications
  MyOPCGroup.IsSubscribed = True
  Exit Sub
ErrorHandler:
  MsgBox "Error: " & Err.Description, vbCritical, "ERROR"
End Sub
```

Figure 80: VBA program to connect to an OPC server in Excel using Siemens OPC DAAutomation 2.0 add-on

(Refer to Appendix III Program Code for the VBA program)

Figure 81 shows the WinCC OPC tag access interface in Excel. Two buttons were created and linked to the Macros for connecting and disconnecting from WinCC OPC server dynamically.

Name	Value	Data Type	Overwrite
MoellerLevelLow	FALSE	Bool	
MoellerLevelHigh	FALSE	Bool	
MoellerTemperatureLow	FALSE	Bool	
MoellerTemperatureHigh	FALSE	Bool	
MoellerPressureLow	FALSE	Bool	
MoellerPressureHigh	FALSE	Bool	
MoellerFeedback	TRUE	Bool	
SEWFaultStatus	0	DInt	
SEWOperatingHours	0	DInt	
SEWCTWControllerInhibit	FALSE	Bool	
SEWCTWRapidStop	TRUE	Bool	
SEWCTWStop	TRUE	Bool	
SEWCTWHoldControl	TRUE	Bool	
SEWCTWRampSet	TRUE	Bool	
SEWCTWParameterSet	TRUE	Bool	
SEWCTWReset	FALSE	Bool	
SEWCTWDIEnable	TRUE	Bool	
SEWCTWDICW	TRUE	Bool	
SEWCTWDICCW	FALSE	Bool	
SEWCTWDISP1	FALSE	Bool	
SEWCTWDISP2	FALSE	Bool	
SEWCTWDIRampSwitch	FALSE	Bool	
SEWCTWDIParameterSwitch	FALSE	Bool	
SEWCTWDIErrorReset	FALSE	Bool	
SEWActualMaximumSpeed	3000	Int	
SEWActualSetpointSpeed	100	Int	
SEWSTWOutputStage	TRUE	Bool	
SEWSTWInverterReady	TRUE	Bool	
SEWSTWPODData	TRUE	Bool	
SEWSTWRampSet	TRUE	Bool	
SEWSTWParameterSet	TRUE	Bool	
SEWSTWFaultWarning	FALSE	Bool	
SEWSTWCWLimitSwitch	FALSE	Bool	
SEWSTWCCWLimitSwitch	FALSE	Bool	
SEWSTWDOReady	TRUE	Bool	
SEWSTWDOOperation	TRUE	Bool	
SEWSTWDOFault	TRUE	Bool	
SEWSTWDORotatingField	TRUE	Bool	
SEWSTWDOSPattern	TRUE	Bool	
SEWSTWDOSpeedReference	FALSE	Bool	
SEWSTWDOSPEqualActualSpeed	TRUE	Bool	
SEWSTWDOMaxCurrent	TRUE	Bool	
SEWActualSpeed	2999	Int	
SEWActualSpeedPercentage	100.0061	Real	
SEWFeedback	FALSE	Bool	
HMIHeartBeat	TRUE	Bool	
DPPACouplerCurrent	38	Real	
DPPACouplerCurrentStatus	TRUE	Bool	
WarningWord	5120	Word	
ErrorWord	2	Word	
WarningStatus	1.01E+09	DWord	
ErrorStatus	131074	DWord	

Name	Value	Data Type	Overwrite
DanfossParameterNumber	342	Bool	
DanfossRequest	1	Int	
DanfossIndex	0	Bool	
DanfossActualPVARead	100	Bool	
DanfossActualPVAWrite	0	Bool	
DanfossCTWReference1	FALSE	Bool	
DanfossCTWReference2	FALSE	Bool	
DanfossCTWDCBrake	TRUE	Bool	
DanfossCTWCoasting	TRUE	Bool	
DanfossCTWQuickStop	TRUE	Bool	
DanfossCTWHoldOutputFrequency	TRUE	Bool	
DanfossCTWStart	TRUE	Bool	
DanfossCTWReset	FALSE	Bool	
DanfossCTWJog	FALSE	Bool	
DanfossCTWRamp	FALSE	Bool	
DanfossCTWControlData	TRUE	Bool	
DanfossCTWRelay01	FALSE	Bool	
DanfossCTWRelay04	FALSE	Bool	
DanfossCTWParameterSet1	FALSE	Bool	
DanfossCTWParameterSet2	FALSE	Bool	
DanfossCTWReverse	FALSE	Bool	
DanfossActualReference	0	Real	
DanfossSpeedLowLimit	0	Int	
DanfossSpeedHighLimit	3000	Int	
DanfossRampUpTime	1	Real	
DanfossRampDownTime	1	Real	
DanfossTorqueLimit	50	Real	
DanfossDO29Set	FALSE	Bool	
DanfossDO27Set	FALSE	Bool	
DanfossRelay1Set	FALSE	Bool	
DanfossRelay2Set	FALSE	Bool	
DanfossPulseOutput	0	Real	
DanfossAnalogOutput	0	Real	
DanfossSTWControlReady	TRUE	Bool	
DanfossSTWDriveReady	TRUE	Bool	
DanfossSTWEnable	TRUE	Bool	
DanfossSTWTrip	FALSE	Bool	
DanfossSTWError	FALSE	Bool	
DanfossSTWTripLock	FALSE	Bool	
DanfossSTWWarning	FALSE	Bool	
DanfossSTWSpeed&Reference	TRUE	Bool	
DanfossSTWControl	TRUE	Bool	
DanfossSTWFrequencyLimit	TRUE	Bool	
DanfossSTWInOperation	TRUE	Bool	
DanfossSTWOvertemperature	FALSE	Bool	
DanfossSTWVoltageExceed	FALSE	Bool	
DanfossSTWTorqueExceed	FALSE	Bool	
DanfossSTWTimerExceed	FALSE	Bool	

Name	Value	Data Type	Overwrite
DanfossMainActualValue	0.00E+00	Int	
DanfossMotorPower	0	Int	
DanfossMotorVoltage	0	Real	
DanfossFrequency	0	Real	
DanfossSpeed	0	Int	
DanfossMotorCurrent	0	Real	
DanfossHeatsinkTemperature	27	Int	
DanfossDI33Status	FALSE	Bool	
DanfossDI32Status	FALSE	Bool	
DanfossDI29Status	FALSE	Bool	
DanfossDI27Status	FALSE	Bool	
DanfossDI19Status	FALSE	Bool	
DanfossDI18Status	FALSE	Bool	
DanfossDO29Status	FALSE	Bool	
DanfossDO27Status	FALSE	Bool	
DanfossFeedback	TRUE	Bool	
EndressTankLevel	57	Real	
EndressTankLevelStatus	TRUE	Byte	
EndressTankDistance	603.9999	Real	
EndressTankDistanceStatus	TRUE	Byte	
EndressColdLevel	60	Real	
EndressColdLevelStatus	TRUE	Byte	
EndressColdTemperature	25.75	Real	
EndressColdTemperatureStatus	TRUE	Byte	
EndressHotLevel	39.5	Real	
EndressHotLevelStatus	TRUE	Byte	
EndressHotTemperature	25.37499	Real	
EndressHotTemperatureStatus	TRUE	Byte	
PIDLevelSP	40	Real	
PIDLevelPV	57	Real	
PIDLevelMV	50	Real	
PIDLevelManualOn	FALSE	Bool	
PIDLevelManualValue	50	Real	
PIDLevelK	30	Real	
PIDLevelTi	1000	Real	
PIDLevelTd	1000	Real	
PIDTemperatureSP	30	Real	
PIDTemperaturePV	31.65	Real	
PIDTemperatureMV	30	Real	
PIDTemperatureManualOn	FALSE	Bool	
PIDTemperatureManualValue	50	Real	
PIDTemperatureK	10	Real	
PIDTemperatureTi	5000	Real	
PIDTemperatureTd	1000	Real	
PIDPressureSP	30	Real	
PIDPressurePV	63.725	Real	
PIDPressureMV	0	Real	
PIDPressureManualOn	FALSE	Bool	
PIDPressureManualValue	50	Real	
PIDPressureK	1	Real	
PIDPressureTi	2000	Real	
PIDPressureTd	1000	Real	

Computer Name
EEPROJ-01

CONNECT DISCONNECT

Last Access Time
10:55:51

Figure 81: Excel interface to access WinCC OPC tags

OPC Communication

Since both WinCC and LabVIEW can be used as an OPC server or OPC client, there are multiple ways to perform the data exchange. The selection is quite important because OPC communication can cause a significant delay while transferring the data. Two pairs of schemes are listed in **Table 15**.

Data transfer direction	Data exchange schemes
WinCC to LabVIEW	WinCC OPC client writes to LabVIEW OPC server.
	LabVIEW OPC client reads from WinCC OPC server.
LabVIEW to WinCC	LabVIEW OPC client writes to WinCC OPC server.
	WinCC OPC client reads from LabVIEW OPC server.

Table 15: Possible OPC data exchange schemes

A few tests were performed and it was confirmed that the minimum delay occurs when LabVIEW is used as an OPC server and WinCC acts as an OPC client.

WinCC OPC Communication Driver

The WinCC OPC communication driver enables WinCC to access data from LabVIEW shared variable as an OPC client. The accessed LabVIEW shared variables are shown in **Figure 82**.

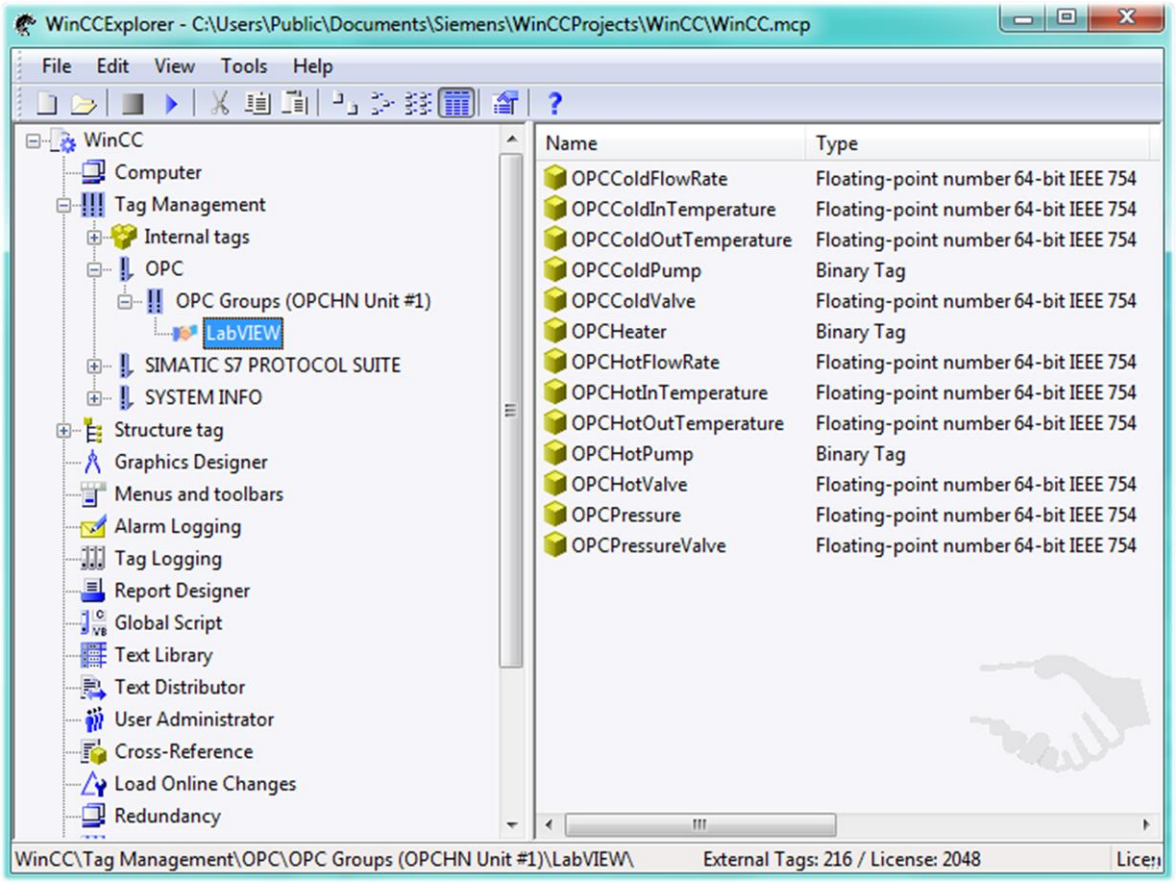


Figure 82: LabVIEW shared variables under OPC communication driver in WinCC

LabVIEW System

Figure 83 shows the structure of the LabVIEW system as part of the whole system.

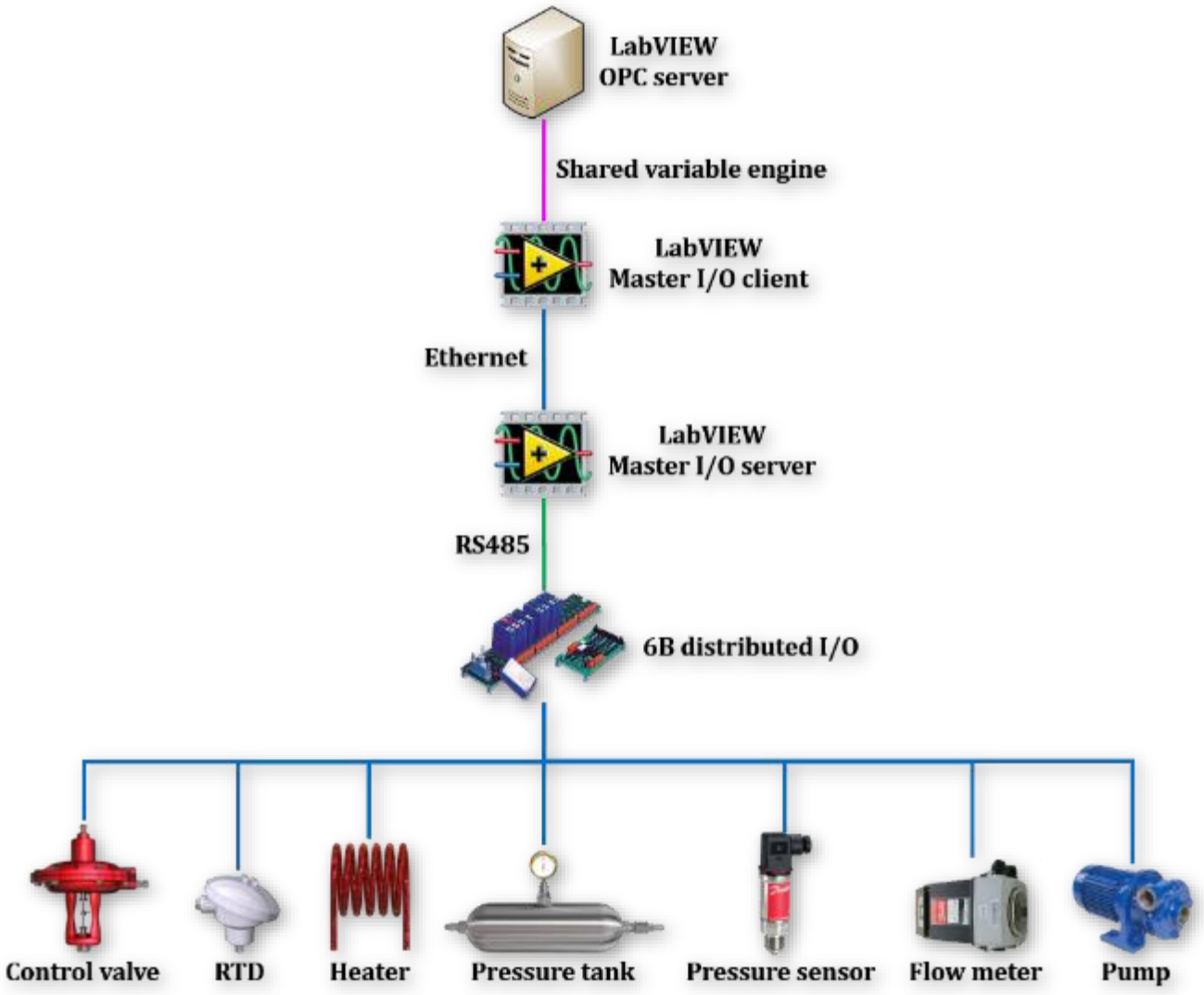


Figure 83: LabVIEW system network structure

LabVIEW Shared Variable Engine

All the network-published shared variables in LabVIEW were recognized by WinCC after deploying them in the project library. Distributed System Manager (DSM) was used as a central monitoring program to organize the shared variable libraries and monitor the health and the traffic of the shared variable engine as shown in **Figure 84**.

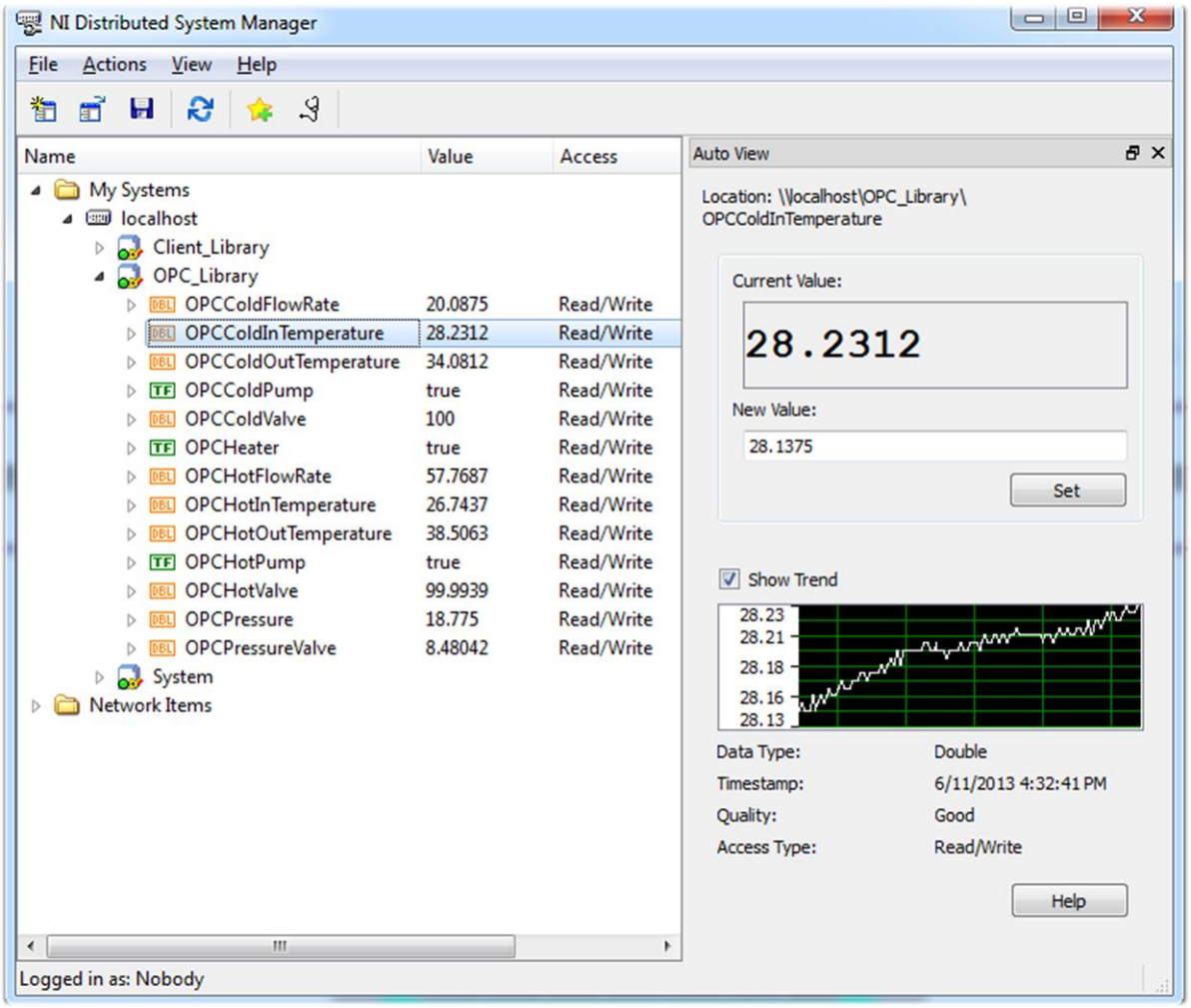


Figure 84: LabVIEW shared variables monitoring in DSM

The Master I/O client program only transferred the process value between WinCC and the Master I/O server program over Ethernet. The Master I/O client program front panel was customized to have a realistic interface to monitor the actual control system in the laboratory as shown in **Figure 85**. The block diagram of the program is relatively simple and is shown in **Figure 86**.

LabVIEW Master I/O Client Program

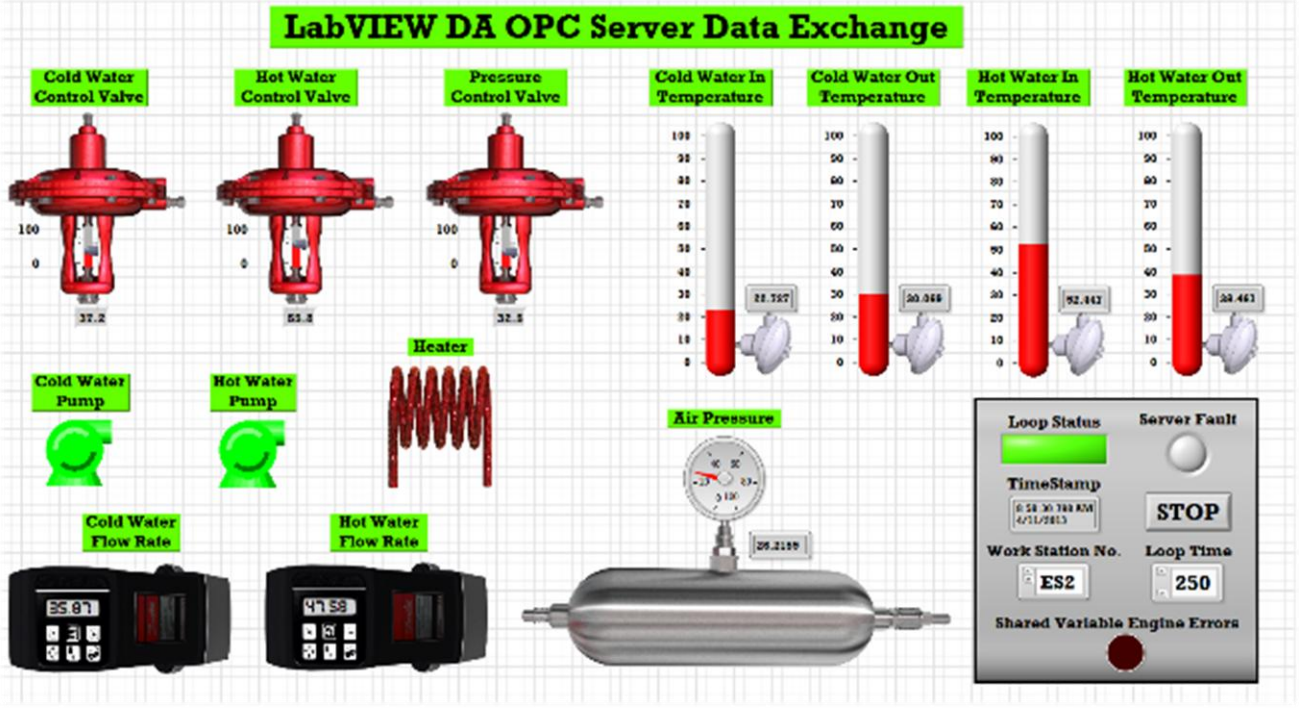


Figure 85: LabVIEW Master I/O client program front panel

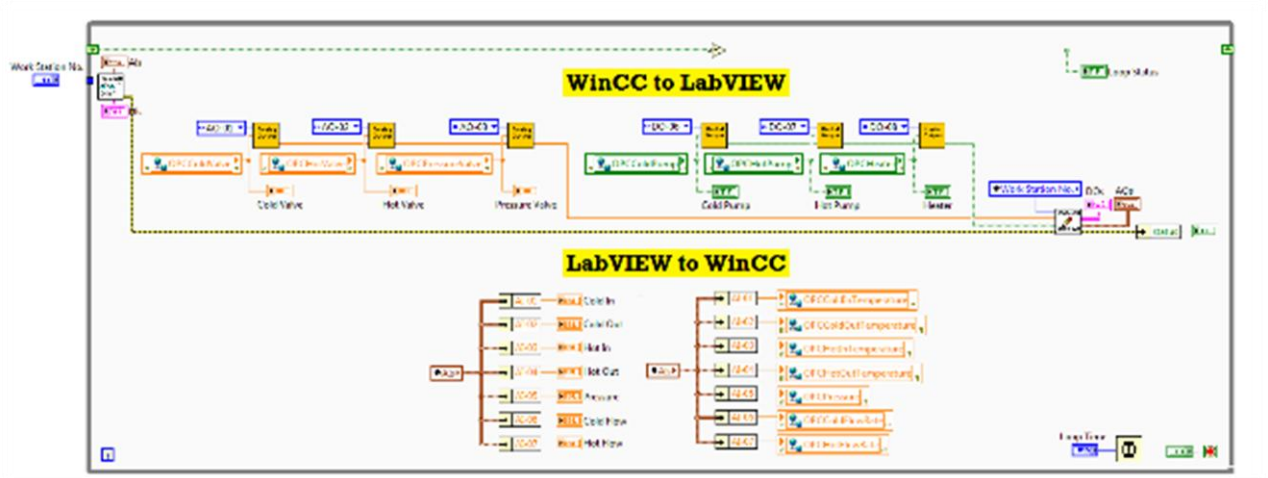


Figure 86: LabVIEW Master I/O client program block diagram

The actual functions of the distributed I/O are summarized in **Table 16**.

Distributed I/O	Functions
AI1	Cold water input temperature.
AI2	Cold water output temperature.
AI3	Hot water input temperature.
AI4	Hot water output temperature.
AI5	Air tank Pressure.
AI6	Cold water flow rate.
AI7	Hot water flow rate.
AO1	Cold water control valve.
AO2	Hot water control valve.
AO3	Pressure control valve.
DO6	Cold water pump.
DO7	Hot water pump.
DO8	Heater.

Table 16: Distributed I/O channels with actual functions

LabVIEW Master I/O Server Program

The Master I/O server program is a communication server for the ICE laboratory that sends the control value to the instruments and takes the measurements from the sensors. The client program was used parallel with the Master I/O server program to implement the control system in the ICE laboratory.

As the actual system was very sensitive, due to the small dimension and volume compared to the significant communication delay, all the unused I/O were deactivated in the Master I/O server program activation table to achieve a faster loop time in order to compensate the communication delay. **Figure 87** is the front panel of LabVIEW Master I/O activation table.

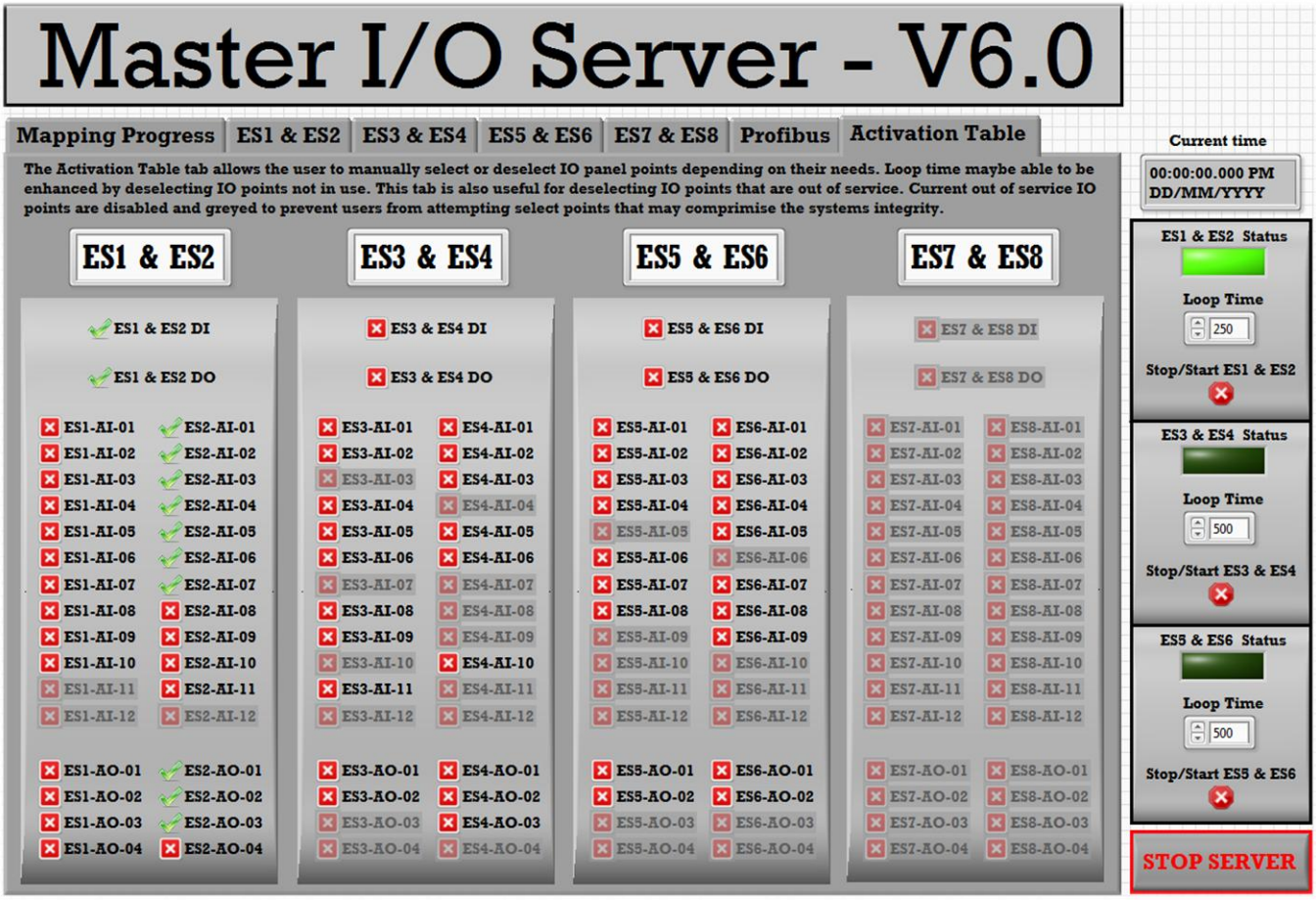


Figure 87: LabVIEW Master I/O server program front panel activation table

NI 6B Distributed I/O Module

The LabVIEW Master I/O server program communicated via the field devices with the 6B distributed I/O modules over RS485 communication. The analog module read the pressure and temperature and sent values to the control valves whereas the digital module switched pumps on and off.

Laboratory Instruments

The ICE laboratory has a range of instruments and they can be freely assembled to build various control systems. The instruments used for the actual control system were 2 water control valves with flow meters, 1 gas control valve with air tank and pressure transmitter, 2 pumps and 4 RTDs.

WinCC System

The network structure of the WinCC system is shown in **Figure 88**.

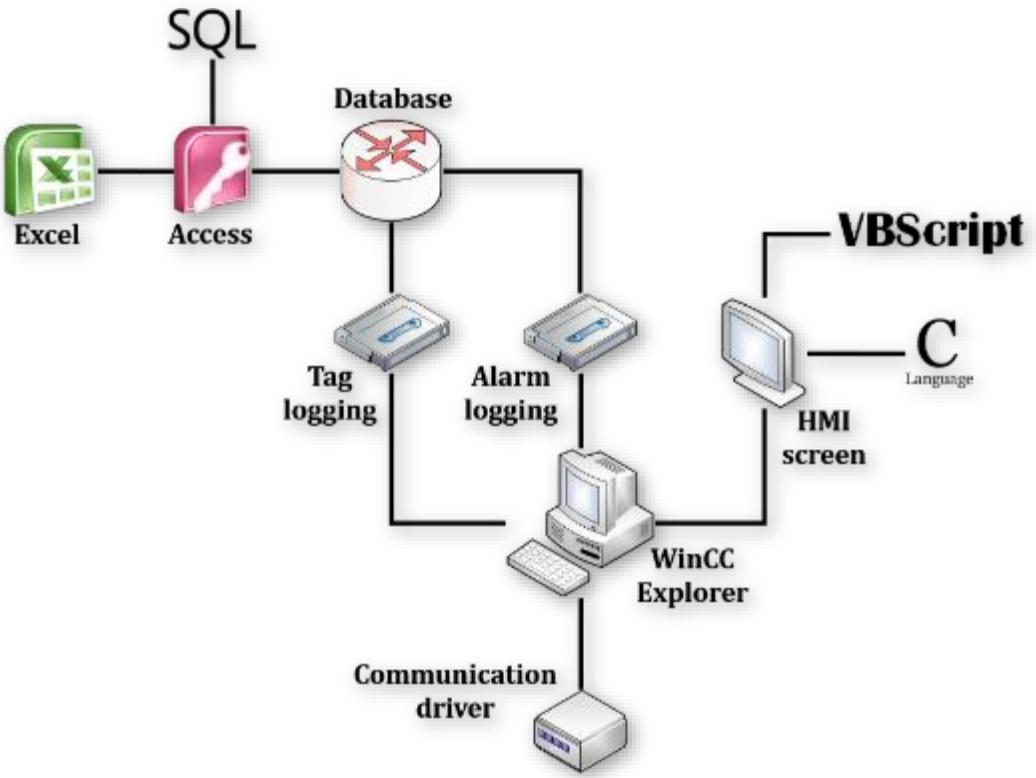


Figure 88: WinCC system network structure

HMI Design

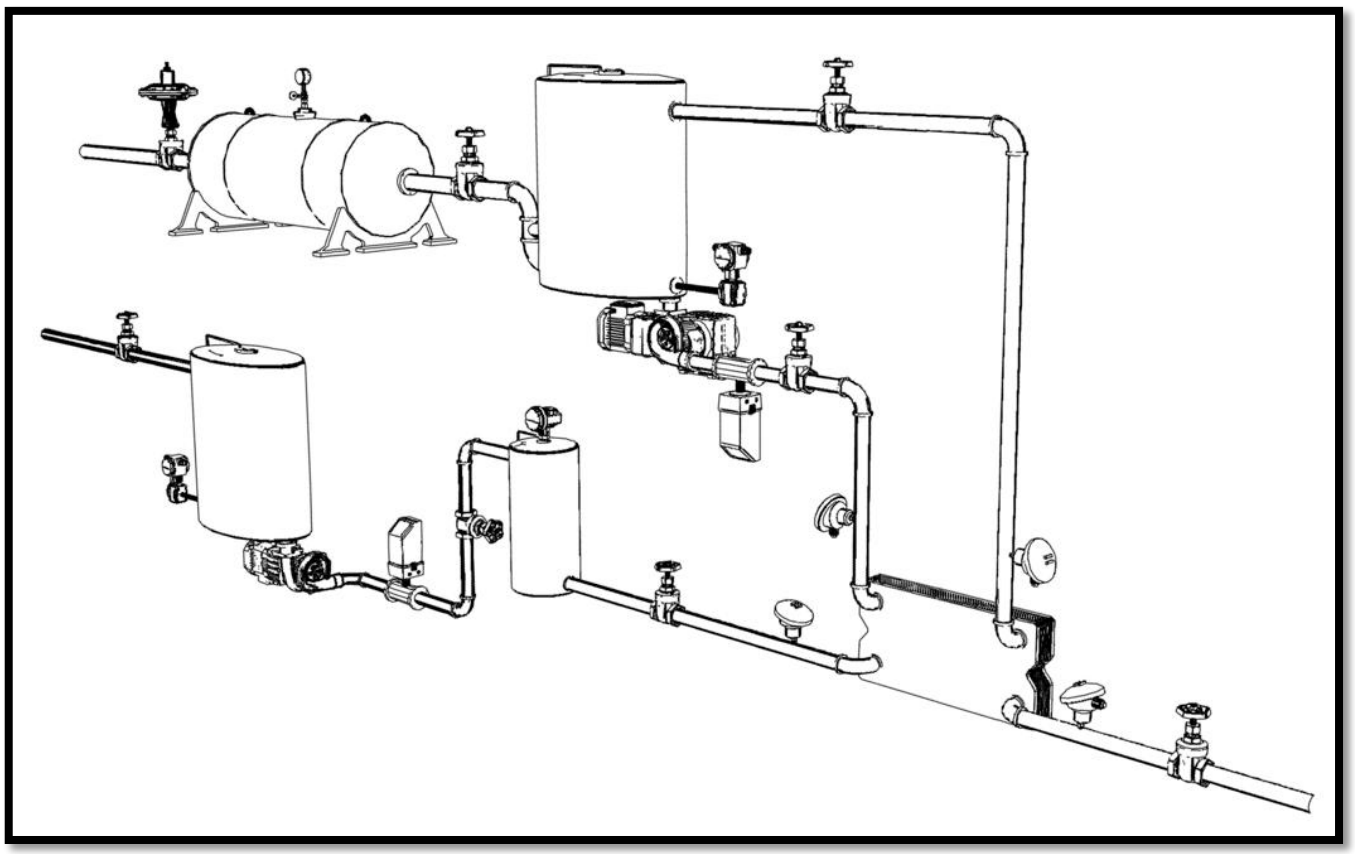


Figure 89: WinCC process HMI interface outlines

The WinCC HMI interface was designed using 3D applications. 3ds Max created 3D models for the instruments (**Figure 89**), Photoshop created textures for the 3D models and finally After Effects performed the colour correction for the rendered graphics. **Figure 90** demonstrates this procedure.

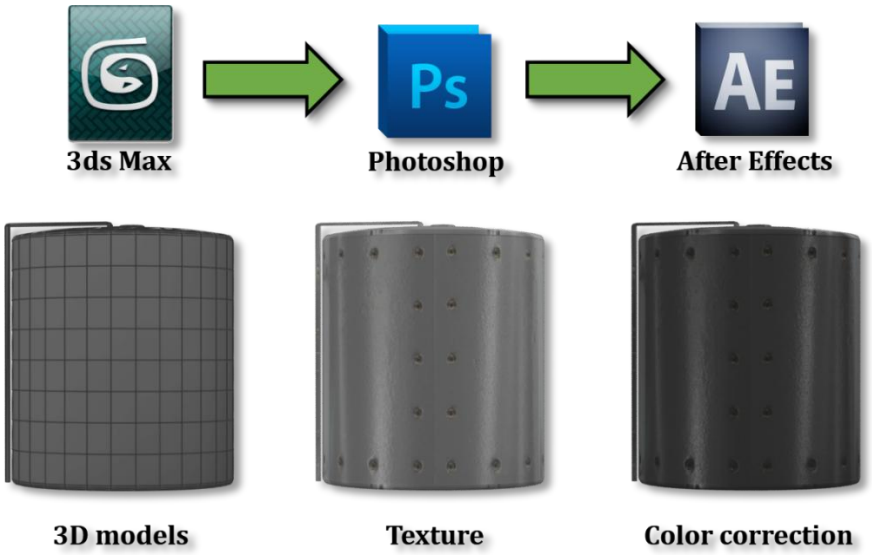


Figure 90: WinCC HMI interface design procedures

Screen

A few screens were created to provide a visualization of the process. A picture object was created to load other screens within the process screen.

Process Screen

The process screen contains all the instruments in the control system with their current values displayed next to them as shown in **Figure 91**. The colour of the value represents different value ranges and clicking each icon underneath will navigate to a certain screen. Whenever a process warning exists, the warning icon will appear and flash.

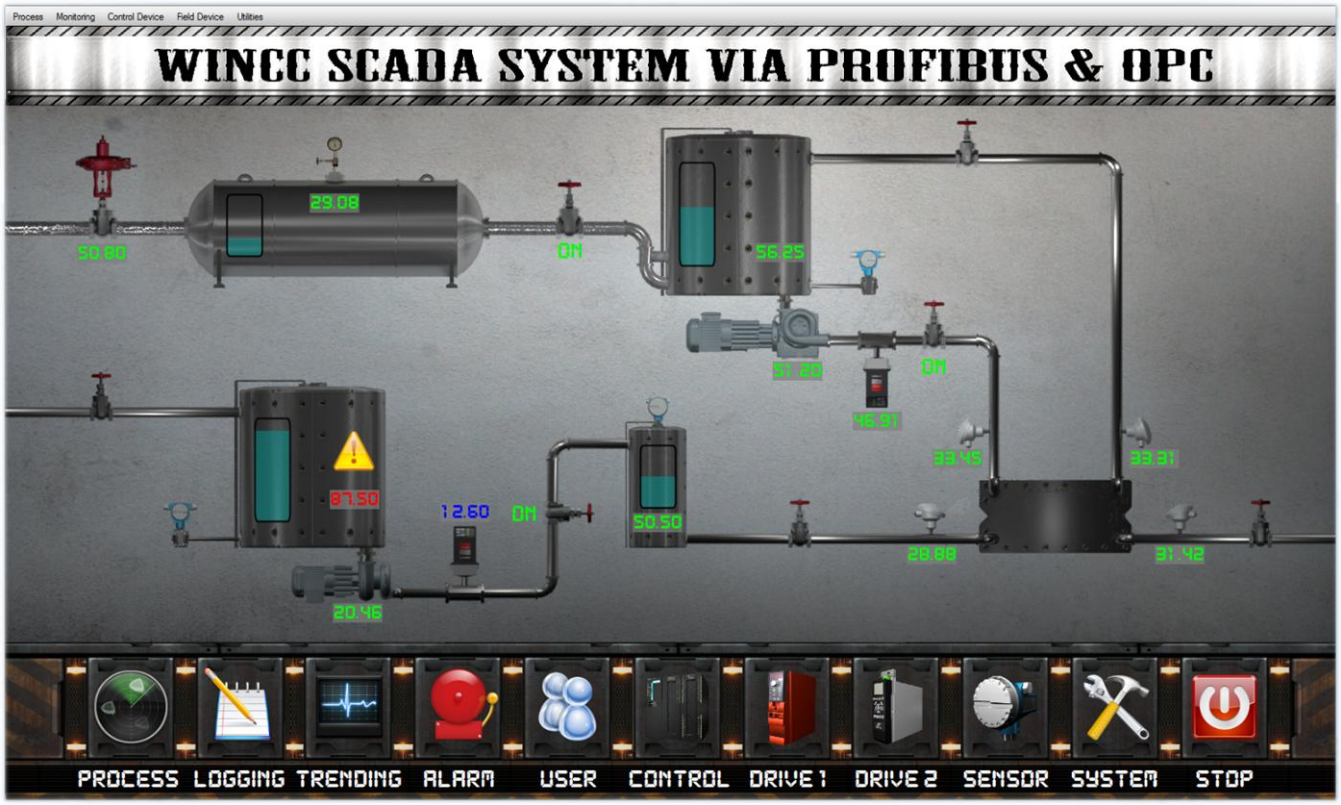


Figure 91: WinCC process screen

Logging Screen

The logging screen (**Figure 92**) cyclically retrieves values for the pre-assigned tags. The displayed values will be stored in the database.

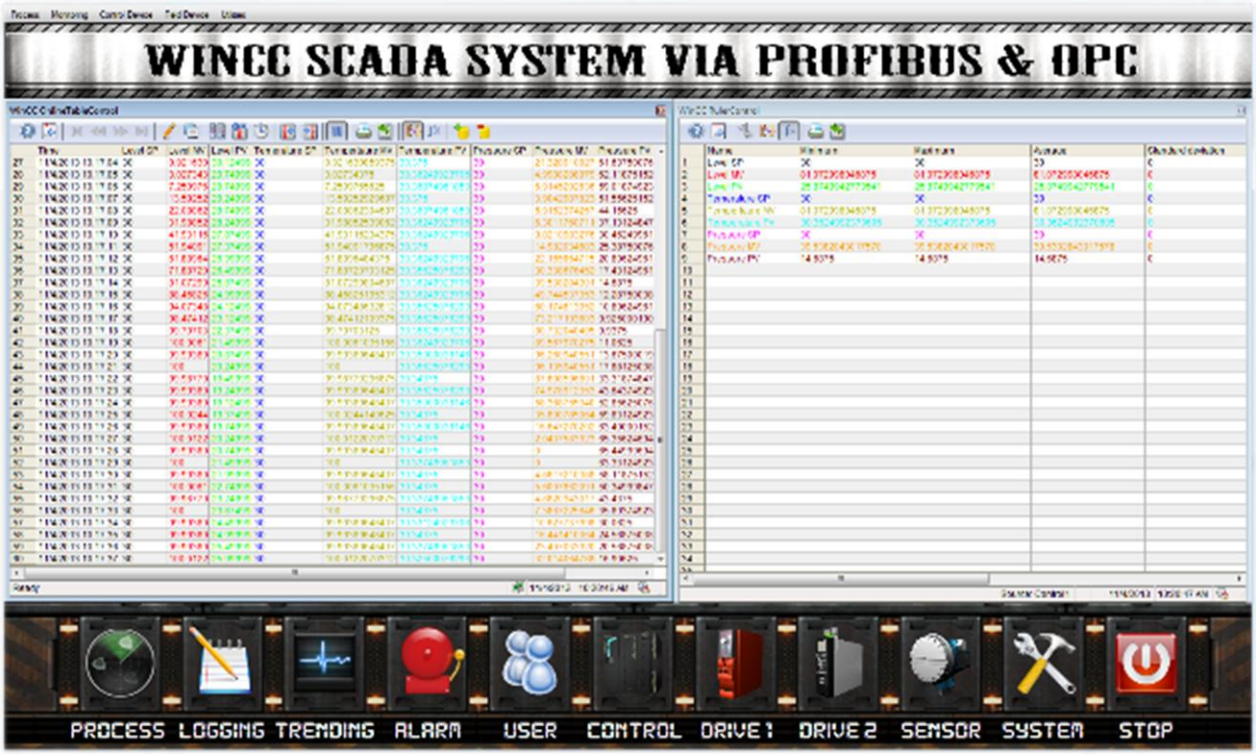


Figure 92: WinCC logging screen

Trending Screen

The trending screen has 4 parts. The first 3 parts are the actual trending for each of the control loops, so the operator does not get confused with a bunch of trends. The 4th part is the statistics window which is used to determine the minimum, maximum and the standard deviation value for a defined period as shown in **Figure 93**.

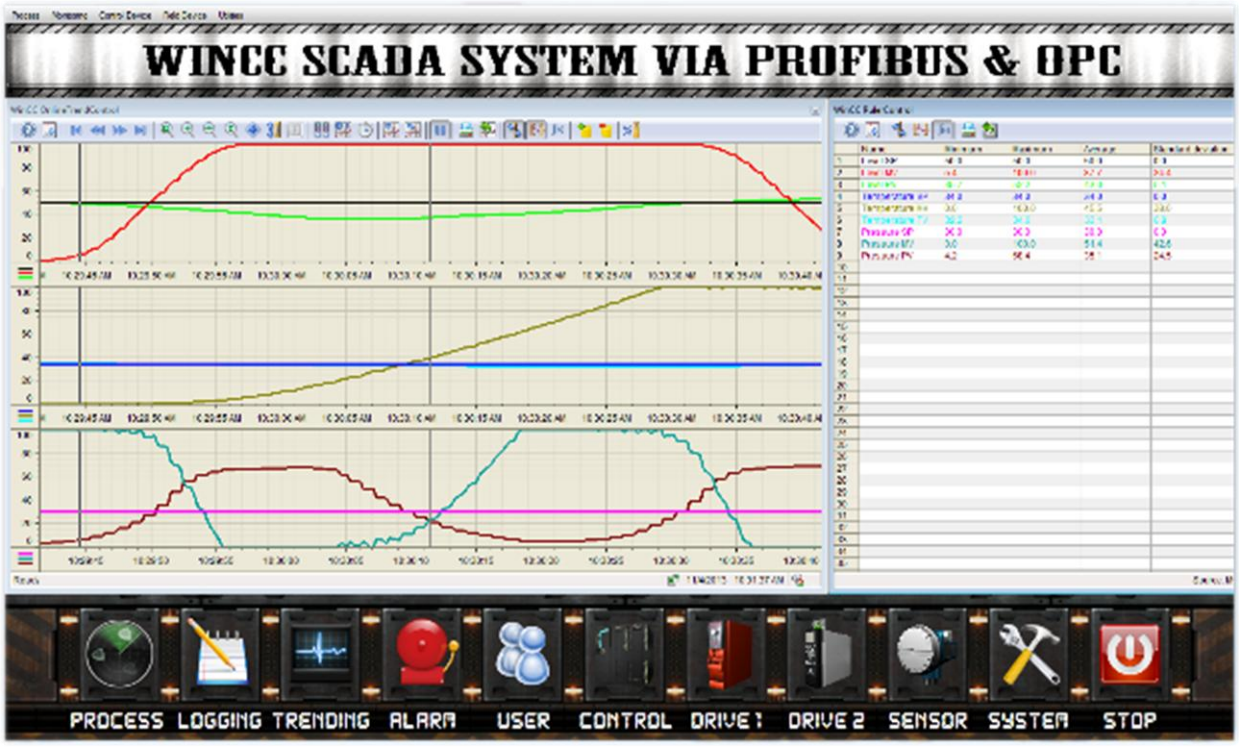


Figure 93: WinCC trending screen

Alarm Screen

The alarm screen (**Figure 94**) basically displays all the active alarms and user can acknowledge them and view the historical alarms.

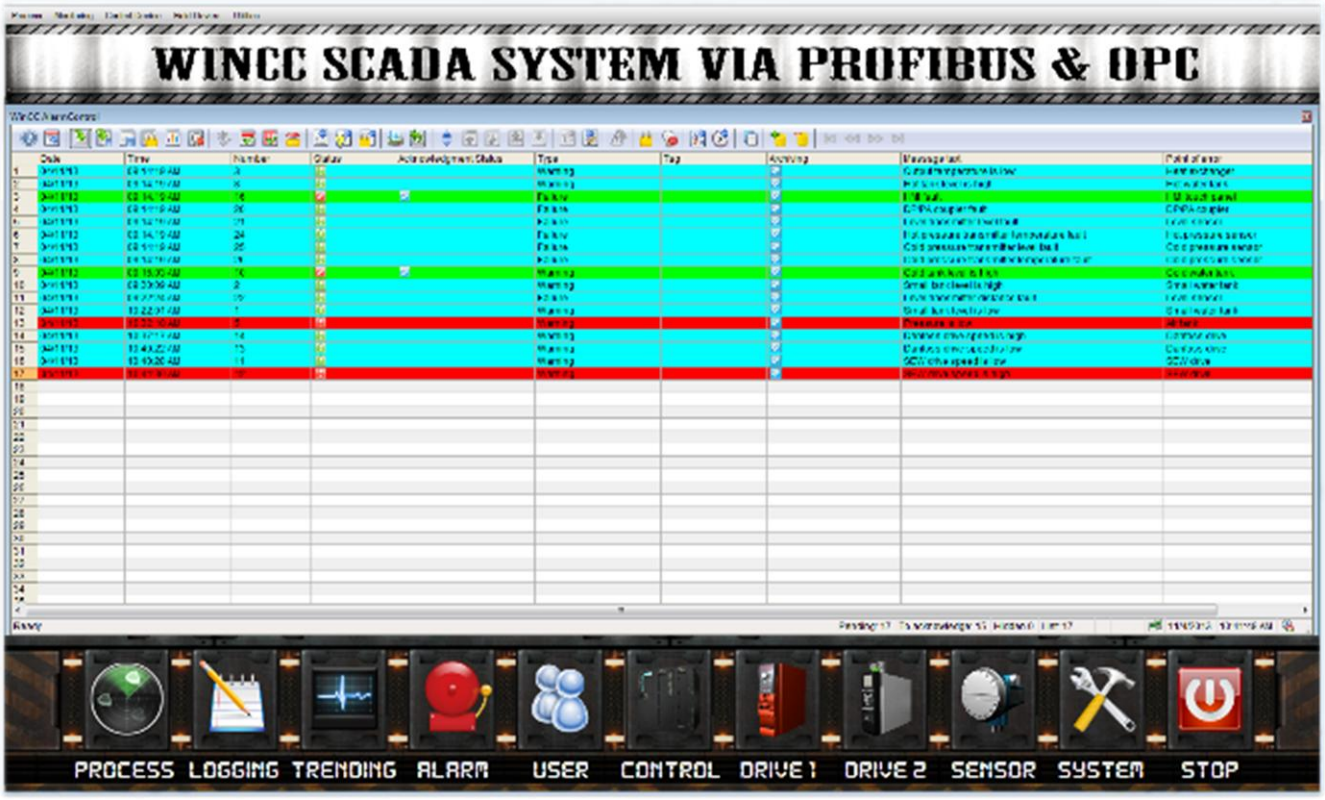


Figure 94: WinCC alarm screen

User Screen

The user screen allows users to log in and log out as shown in **Figure 95**.

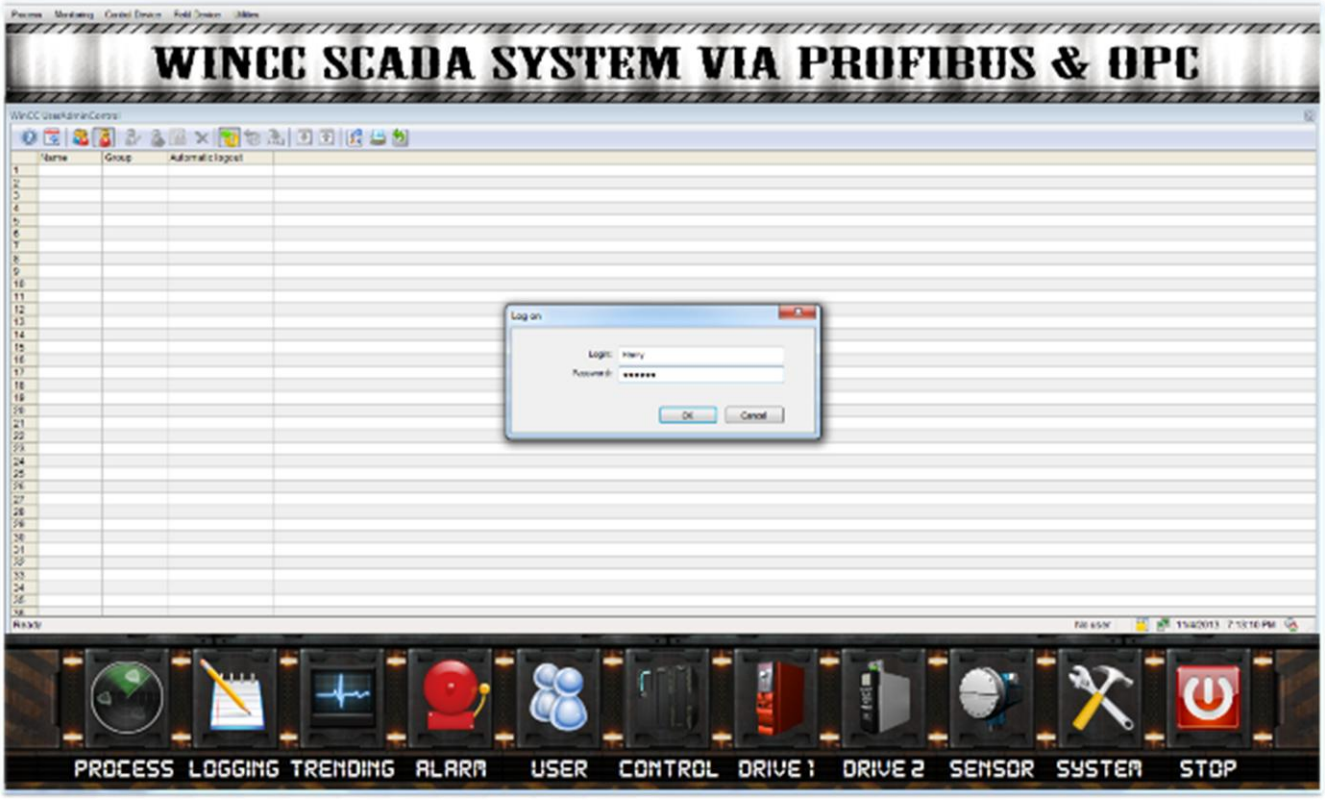


Figure 95: WinCC user screen

Control Screen

The control screen is used to monitor the condition and status of the PLC, TP 177B 6" HMI touch panel and the EASY719 controller. It also provides value entry boxes for operator to set tuning parameters for individual control loops and dynamically swap between manual and auto control as shown in **Figure 96**.

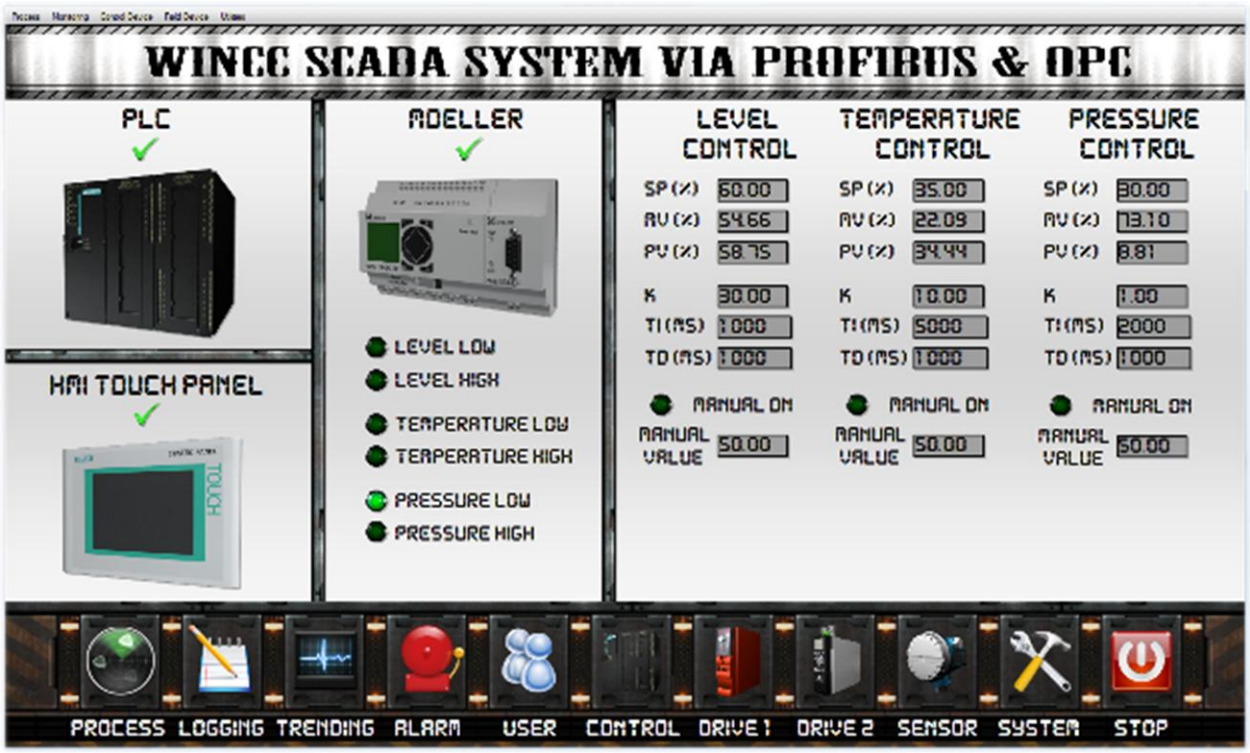


Figure 96: WinCC control screen

SEW Drive Screen

The operator can modify the control word and monitor the status word in the SEW drive screen. The text button of the control word can be pressed to toggle the state of an individual bit in the control word. In addition, the operating hours and the status of the gateway are also monitored here as shown in **Figure 97**.

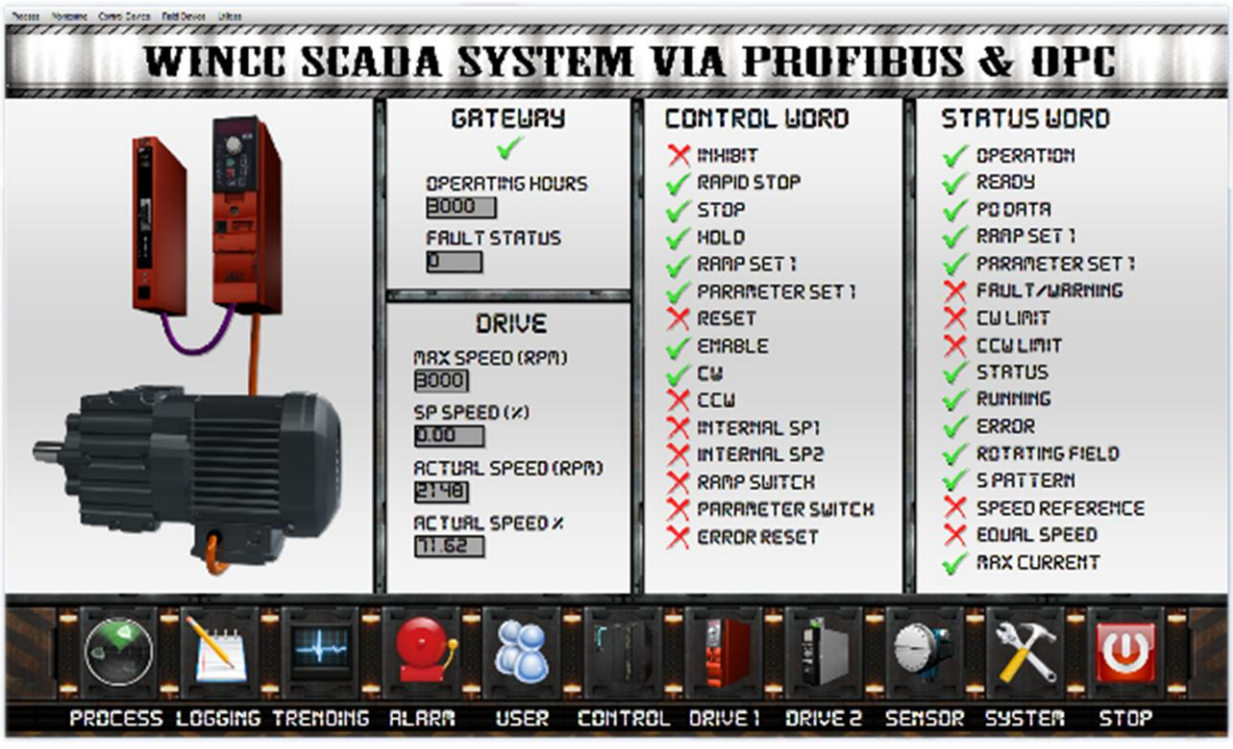


Figure 97: WinCC SEW screen

Danfoss Drive Screen

Similar to the SEW drive screen, the Danfoss drive screen also has the toggle function to change the control word. The actual digital I/O are monitored as well along with a range of drive and motor parameters such as voltage and temperature, as shown in **Figure 98**. In this screen, users can also access the parameters of the Danfoss drive.

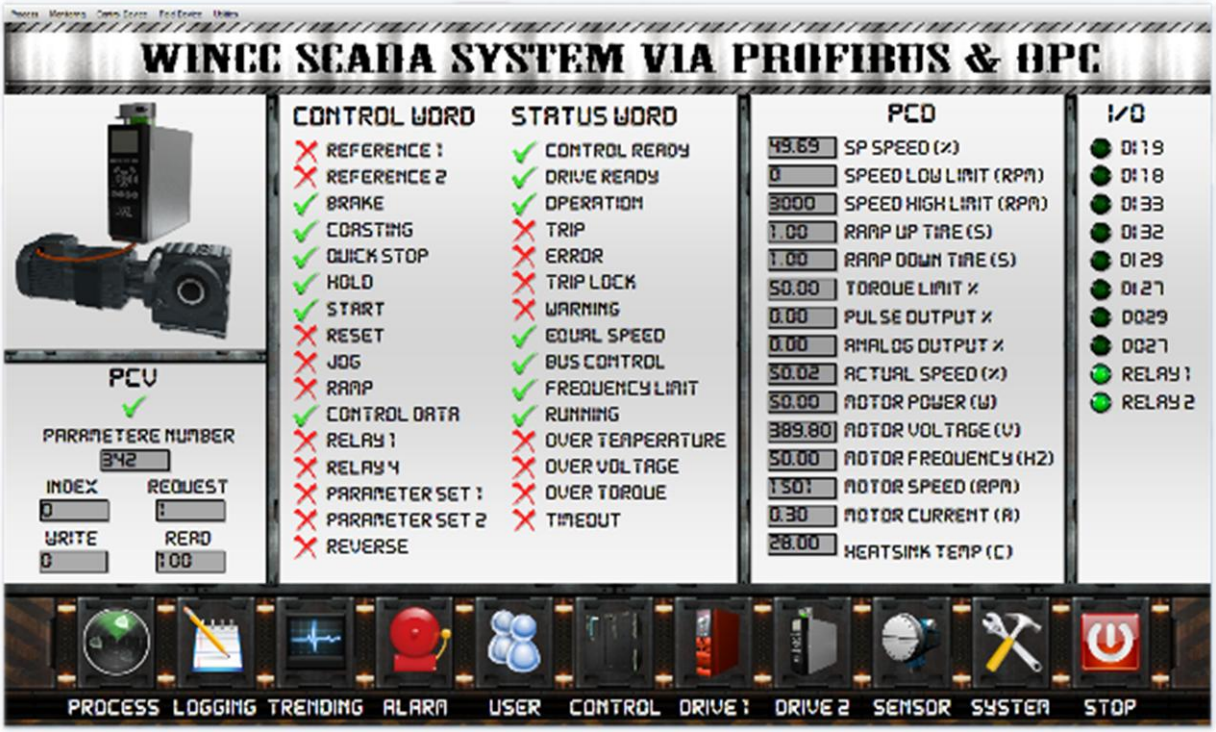


Figure 98: WinCC Danfoss screen

Transmitter Screen

The transmitter screen contains the slave module status of PA devices that includes the DP/PA coupler, a level transmitter and 2 pressure transmitters (**Figure 99**). The actual values of the transmitter measurements are also calculated from the PLC and displayed here.

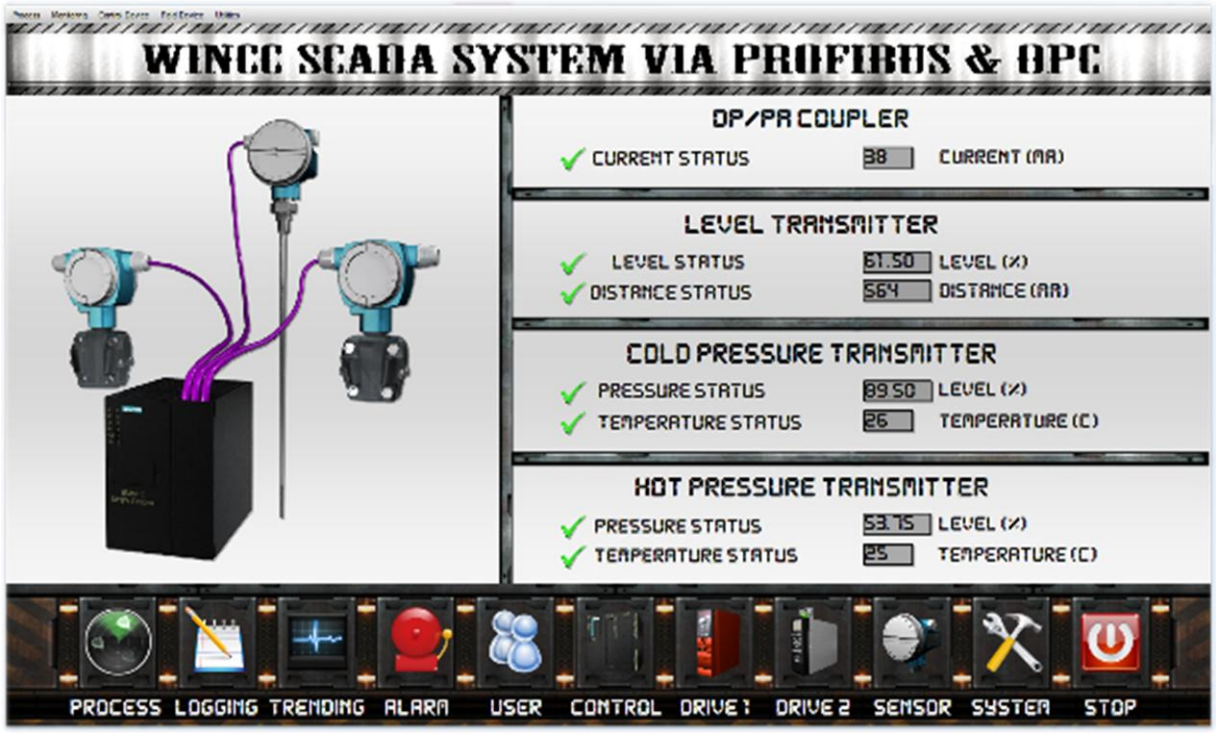


Figure 99: WinCC Transmitter screen

System Screen

In the system screen, the diagnosis window continuously shows the condition of all the communication drivers (**Figure 100**). The global script window contains all the ANSI-C actions for transferring data between 2 PLCs and LabVIEW. The operator can activate/deactivate certain scripts from the global script manually. The CPU utilities, free memory and available drive space were also shown here. The script diagnosis window was used to display the feedback information when a script is triggered.

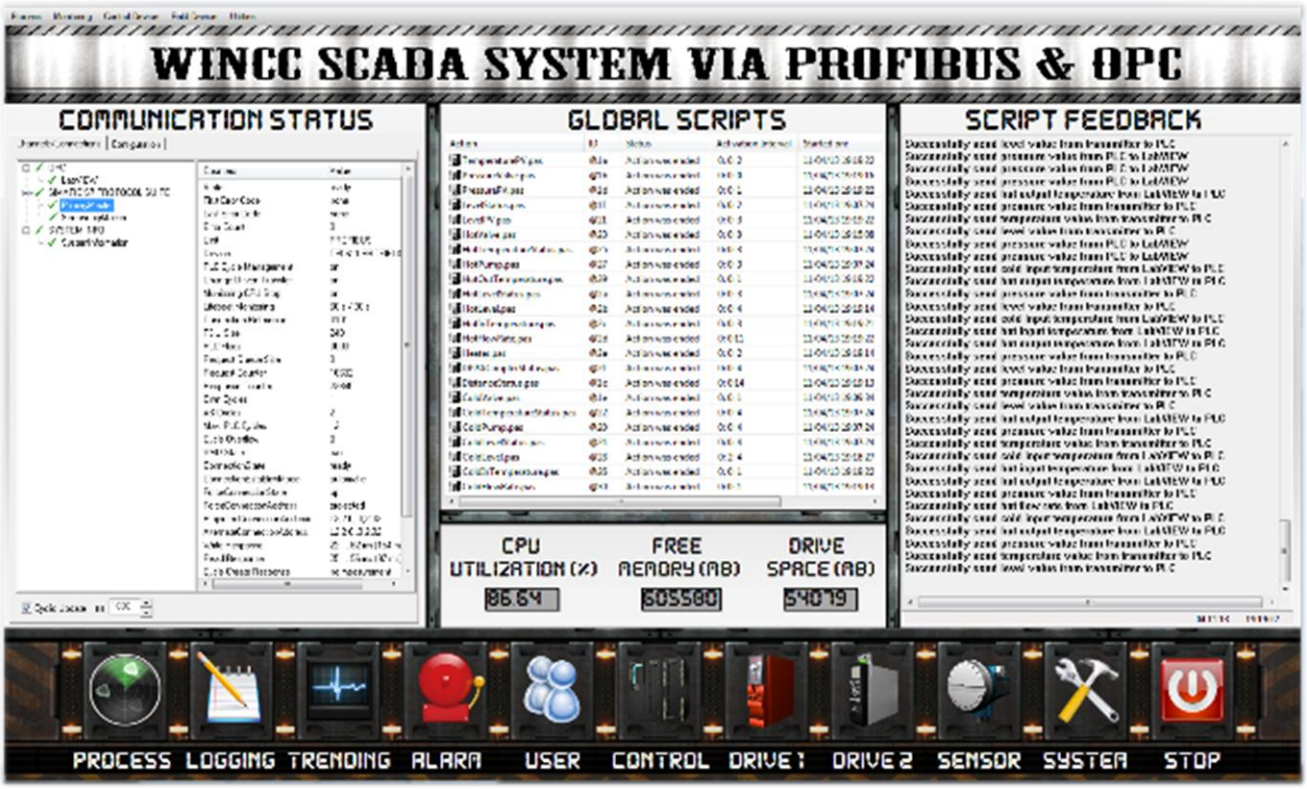


Figure 100: WinCC system screen

(Refer to Appendix XIII WinCC Configuration Instructions for WinCC graphics configuration)

Menu Bar

A tree structured menu bar (**Figure 101**) was created which appears at the top of the Runtime window to allow individual screens to be loaded in a full screen style instead of inside the picture loader object.

Figure 101: Menu bar in WinCC Runtime window

The foundation of the menu bar is the global VBScript (**Figure 102**) and the script was loaded into the **Menus and toolbars** configuration to achieve the dynamics.

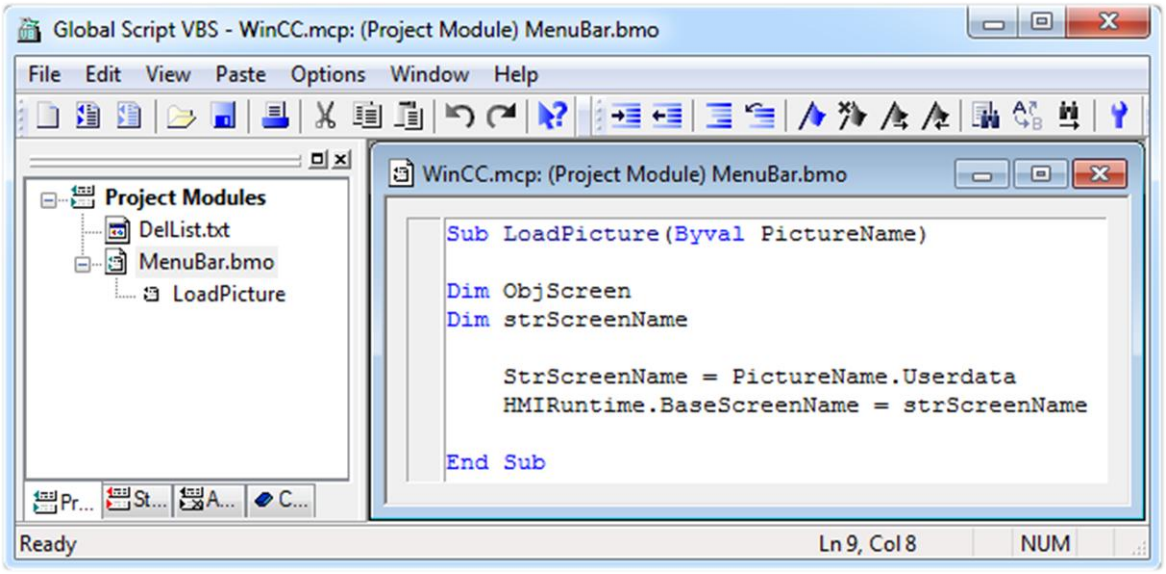


Figure 102: VBScript to create menu bar in global script

(Refer to Appendix XIII WinCC Configuration Instructions for WinCC menu configuration)

Scripting

ANSI-C

Since ANSI-C is good at manipulating large amount of tags, it was primary used to transfer values from the secondary PLC and LabVIEW to the primary PLC. **Figure 103** is the ANSI-C action to transfer data between LabVIEW and PLC.

```
#include "apdefap.h"

int gscAction( void )
{

#pragma option(mbcs)

SetTagFloat("HotOutTemperature",GetTagFloat("OPCHotOutTemperature"));

printf ("Successfully send hot output temperature from LabVIEW to PLC\r\n");

return 0;

}
```

Figure 103: ANSI-C action to transfer data between LabVIEW and PLC

VBS

VBScript is an object oriented language and is therefore very easy to use to handle object properties. It was extensively used in the configuration for animation items such as visibility and colour (**Figure 104**).

```
Function ForeColor_Trigger(Byval Item)

    Dim objtag

    Set objTag = HMIRuntime.Tags("EndressColdLevel")
    objTag.Read

    If objTag.Value < 20 Then
    ForeColor_Trigger = RGB(0,0,255)

    Elseif objtag.Value > 20 And objtag.Value < 80 Then
    ForeColor_Trigger = RGB(0,255,0)

    Else
    ForeColor_Trigger = RGB(255,0,0)

    End If

    HMIRuntime.Trace "Cold storage tank level state changed" & vbCrLf

End Function
```

Figure 104: VBScript to animate colour

(Refer to Appendix III Program Code for WinCC scripts)

Tag Logging

The tag logging was set to record data with a cycle period of 1 second. All the important process values were logged as shown in **Figure 105**.

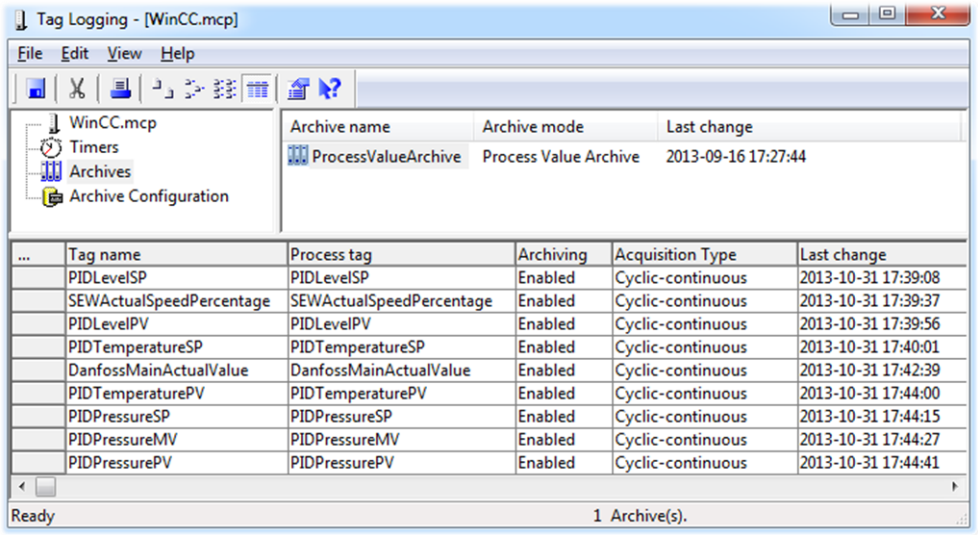


Figure 105: Tag logging configuration in WinCC Explorer

(Refer to Appendix XIII WinCC Configuration Instructions for WinCC tag logging configuration)

Alarm Logging

Two different types of alarms, warning and failure, were integrated in WinCC (**Figure 106**). The alarms were based on the warning word and error word built in the primary PLC. The message text and the point of error were also reported to deliver more information about the process upon errors.

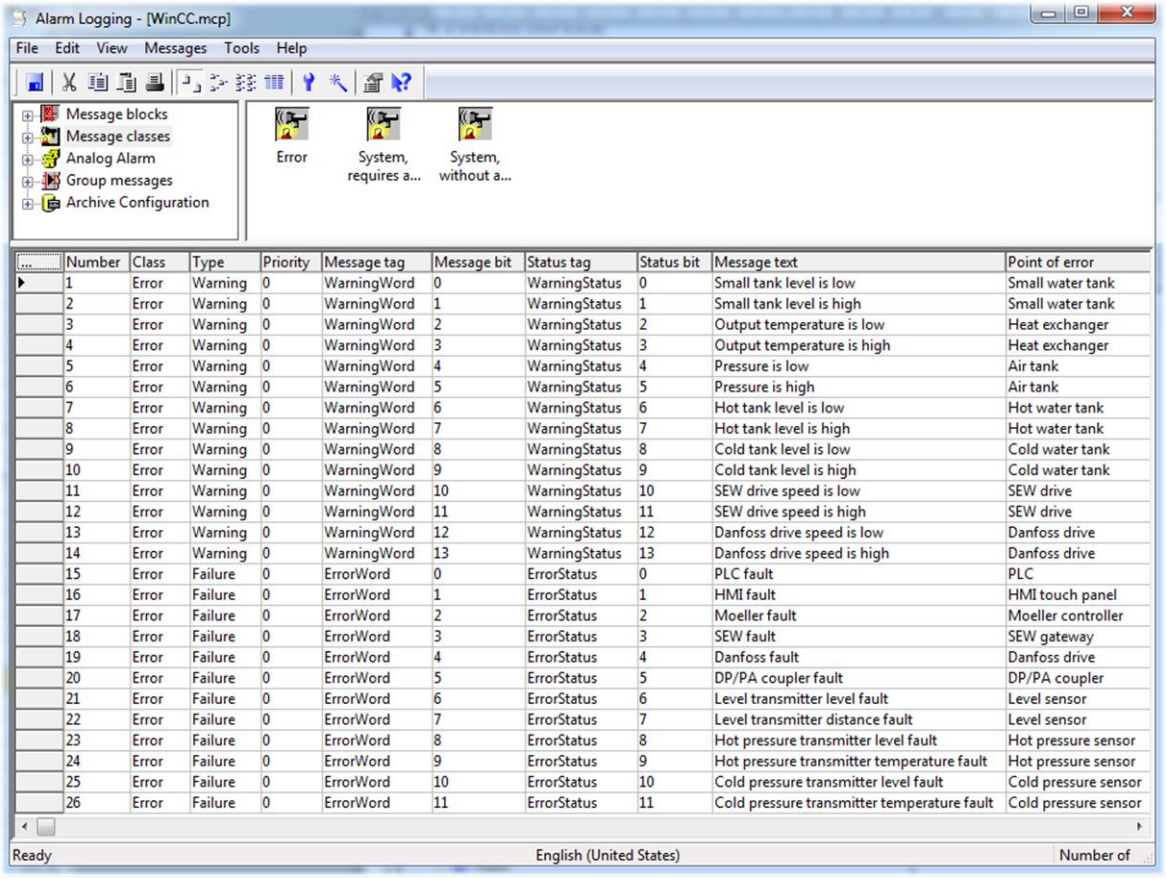


Figure 106: Alarm logging configuration in WinCC Explorer

(Refer to Appendix XIII WinCC Configuration Instructions for WinCC alarm logging configuration)

Access Database Query

An SQL query was made to obtain certain process data from the huge range of the data within the database (**Figure 107**). [103]

Data Table

ID	Level Time	Level SP	Level MV	Level PV	Temperature Time	Temperature SP	Temperature MV	Temperature PV	Pressure Time	Pressure SP	Pressure MV	Pressure PV
1	1:58:10 PM	50	42.11	50.19	1:58:10 PM	30	26.25	29.45	1:58:10 PM	30	48.4	30.18
2	1:58:11 PM	50	42.11	50.22	1:58:11 PM	30	28.91	29.48	1:58:11 PM	30	48.38	30.17
3	1:58:12 PM	50	39.2	50.26	1:58:12 PM	30	28.91	29.51	1:58:12 PM	30	48.38	30.2
4	1:58:13 PM	50	39.2	50.25	1:58:13 PM	30	28.91	29.54	1:58:13 PM	30	48.38	30.2
5	1:58:14 PM	50	39.2	50.24	1:58:14 PM	30	28.91	29.57	1:58:14 PM	30	48.38	30.25
6	1:58:15 PM	50	39.2	50.22	1:58:15 PM	30	28.91	29.58	1:58:15 PM	30	48.38	30.24
7	1:58:16 PM	50	39.2	50.22	1:58:16 PM	30	27.92	29.58	1:58:16 PM	30	48.16	30.22
8	1:58:17 PM	50	39.92	50.24	1:58:17 PM	30	27.92	29.59	1:58:17 PM	30	48.16	30.2
9	1:58:18 PM	50	39.92	50.26	1:58:18 PM	30	27.92	29.57	1:58:18 PM	30	48.16	30.2
10	1:58:19 PM	50	39.92	50.26	1:58:19 PM	30	27.92	29.56	1:58:19 PM	30	48.16	30.23
11	1:58:20 PM	50	39.92	50.23	1:58:20 PM	30	27.92	29.54	1:58:20 PM	30	48.16	30.24
12	1:58:21 PM	50	39.92	50.21	1:58:21 PM	30	28.83	29.52	1:58:21 PM	30	48.35	30.24
13	1:58:22 PM	50	41.13	50.22	1:58:22 PM	30	28.83	29.47	1:58:22 PM	30	48.35	30.25
14	1:58:23 PM	50	41.13	50.22	1:58:23 PM	30	28.83	29.44	1:58:23 PM	30	48.35	30.25
15	1:58:24 PM	50	41.13	50.23	1:58:24 PM	30	28.83	29.42	1:58:24 PM	30	48.35	30.23
16	1:58:25 PM	50	41.13	50.23	1:58:25 PM	30	28.83	29.43	1:58:25 PM	30	48.35	30.23
17	1:58:26 PM	50	41.13	50.2	1:58:26 PM	30	30.3	29.43	1:58:26 PM	30	48.32	30.24
18	1:58:27 PM	50	40.56	50.16	1:58:27 PM	30	30.3	29.46	1:58:27 PM	30	48.32	30.27
19	1:58:28 PM	50	40.56	50.16	1:58:28 PM	30	30.3	29.47	1:58:28 PM	30	48.32	30.27
20	1:58:29 PM	50	40.56	50.18	1:58:29 PM	30	30.3	29.47	1:58:29 PM	30	48.32	30.23
21	1:58:30 PM	50	40.56	50.18	1:58:30 PM	30	30.3	29.39	1:58:30 PM	30	48.32	30.22
22	1:58:31 PM	50	40.56	50.17	1:58:31 PM	30	30.73	29.39	1:58:31 PM	30	48.23	30.22
23	1:58:32 PM	50	40.36	50.13	1:58:32 PM	30	30.73	29.33	1:58:32 PM	30	48.23	30.24
24	1:58:33 PM	50	40.36	50.13	1:58:33 PM	30	30.73	29.3	1:58:33 PM	30	48.23	30.24
25	1:58:34 PM	50	40.36	50.11	1:58:34 PM	30	30.73	29.3	1:58:34 PM	30	48.23	30.24
26	1:58:35 PM	50	40.36	50.12	1:58:35 PM	30	30.73	29.37	1:58:35 PM	30	48.23	30.21
27	1:58:36 PM	50	40.36	50.13	1:58:36 PM	30	31.17	29.41	1:58:36 PM	30	48.25	30.2
28	1:58:37 PM	50	39.09	50.12	1:58:37 PM	30	31.17	29.41	1:58:37 PM	30	48.25	30.18
29	1:58:38 PM	50	39.09	50.07	1:58:38 PM	30	31.17	29.45	1:58:38 PM	30	48.25	30.18
30	1:58:39 PM	50	39.09	50.05	1:58:39 PM	30	31.17	29.49	1:58:39 PM	30	48.25	30.19
31	1:58:40 PM	50	39.09	50.06	1:58:40 PM	30	31.17	29.51	1:58:40 PM	30	48.25	30.19
32	1:58:41 PM	50	39.09	50.08	1:58:41 PM	30	29.92	29.52	1:58:41 PM	30	48.45	30.17
33	1:58:42 PM	50	38.02	50.07	1:58:42 PM	30	29.92	29.54	1:58:42 PM	30	48.45	30.12
34	1:58:43 PM	50	38.02	50.06	1:58:43 PM	30	29.92	29.54	1:58:43 PM	30	48.45	30.12
35	1:58:44 PM	50	38.02	50	1:58:44 PM	30	29.92	29.53	1:58:44 PM	30	48.45	30.14
36	1:58:45 PM	50	38.02	50	1:58:45 PM	30	29.92	29.55	1:58:45 PM	30	48.45	30.13
37	1:58:46 PM	50	38.02	49.98	1:58:46 PM	30	30.51	29.55	1:58:46 PM	30	48.51	30.13
38	1:58:47 PM	50	38.74	50	1:58:47 PM	30	30.51	29.54	1:58:47 PM	30	48.51	30.12
39	1:58:48 PM	50	38.74	49.98	1:58:48 PM	30	30.51	29.54	1:58:48 PM	30	48.51	30.12

Figure 107: Retrieved and sorted data in Access

Excel Controller Performance Analysis

Excel imported the query data from Access and uses appropriate controller performance criteria to analyse the performance of the controller with different tuning parameters (**Figure 108**).

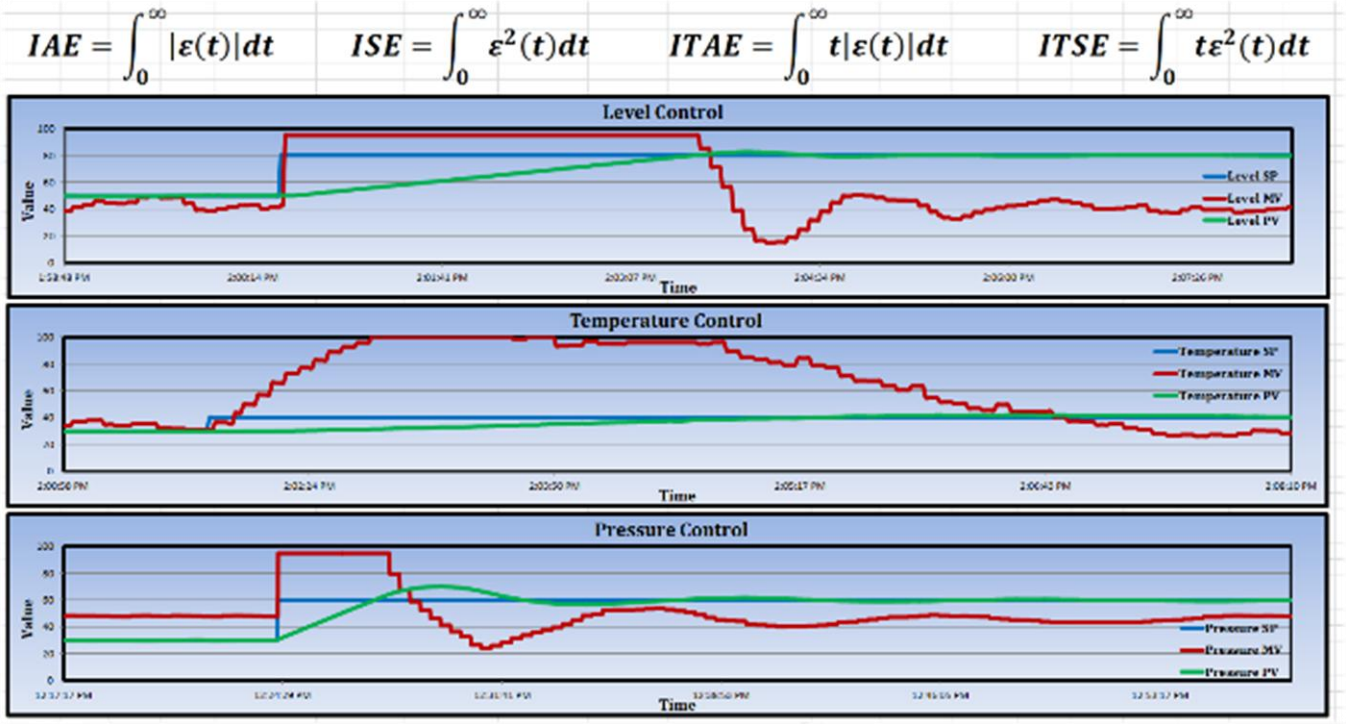


Figure 108: Controller performance analysis in Excel

CHAPTER 8: DOCUMENTATION

One of the major objectives of this project is to create a series of comprehensive instructions which will guide future ICSE students while they are learning SCADA systems and fieldbus communication.

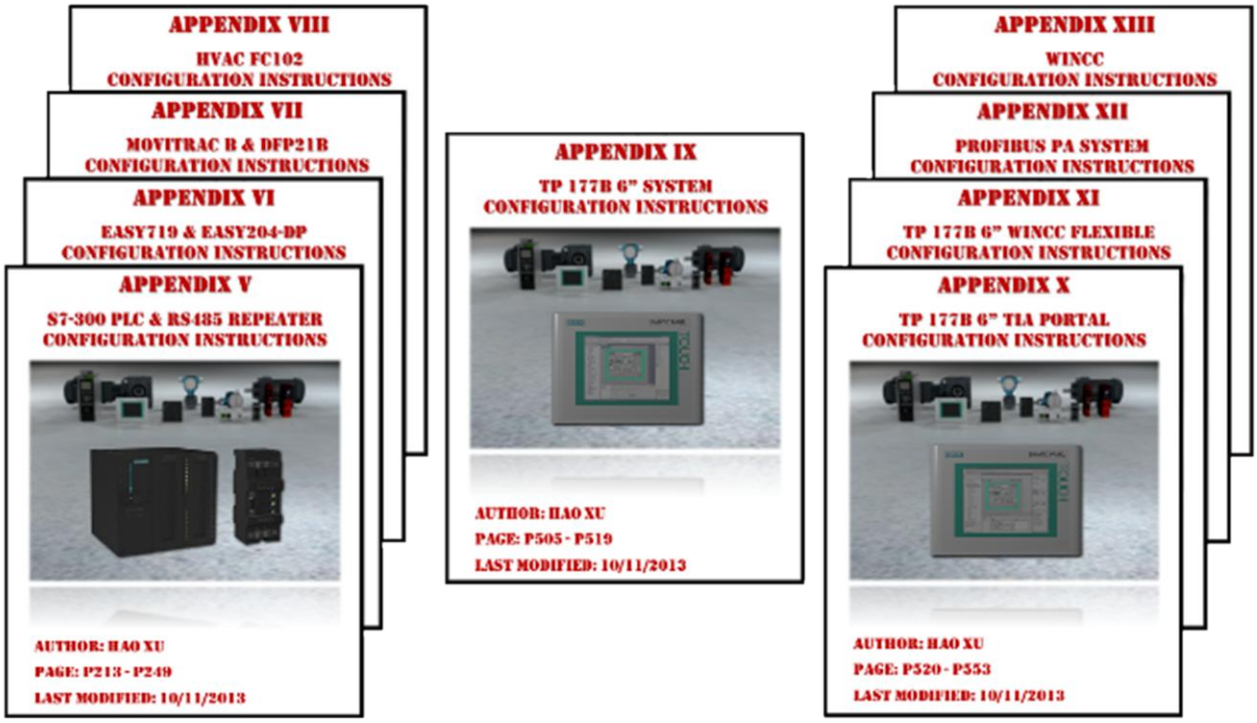


Figure 109: Configuration instructions created

The documentation created covers everything that has been touched during this project and the list of the documentation is as follows:

- **S7-300 PLC & Repeater Configuration Instruction** (PLC specifications, TIA Portal configuration, MS communication and repeater configuration)
- **TP 177B 6" System Configuration Instruction** (HMI touch panel specifications, operation and system configuration)
- **TP 177B 6" TIA Portal Configuration Instruction** (HMI touch panel screen, object and function configuration in TIA Portal)
- **TP 177B 6" WinCC flexible Configuration Instruction** (HMI touch panel screen, object and function configuration in WinCC flexible)
- **Easy719 Configuration Instruction** (EASY719 specification, operations, functions, slave module communication, EASYSOFT configuration and sample code)
- **MOVITRAC B Configuration Instruction** (MOVITRAC B and gateway specifications, MotionStudio configuration, manual & terminal operations, slave module communication and sample code)
- **HVAC FC102 Configuration Instruction** (HVAC FC102 specifications, MCT 10 configuration, manual & terminal operations, slave module communication and sample code)
- **Profibus PA System Configuration Instruction** (DP/PA coupler, Levelflex M FMP40 and Deltabar S PMD70 specifications, slave module communication and sample code)
- **WinCC Configuration Instruction** (Profibus & OPC communication, object, control, logging, and alarm configuration)

Huge amount of time was spent to create clear documentations and every effort was made to ensure the instructions are easy to follow without missing any key information.

(Refer to portfolio in the document for the configuration instructions)

Sample Code

For each type of the slave module of a device, there is a typical online sample code included to demonstrate the structure of the telegram. Since the peripheral memory addresses in TIA Portal vary, each slave module was set to reserve a certain range of the peripheral memory addresses to eliminate the confusion for students who are not familiar with data communication. **Table 17** shows the reversed peripheral memory address for each device.

Devices	Slave modules	Input	Output
EASY719	Input 1 byte	IB256:P	N/A
	Output 1 byte	N/A	QB256:P
	Input 3 byte	IB257:P – IB259:P	N/A
	Output 3 byte	N/A	QB257:P – QB259:P
	EASY 700/800 control commands	IB260:P – IB268:P	QB260:P – QB268:P
DFP21B	Parameter channel	IB269:P – IB276:P	QB269:P – QB276:P
	PD for the 1st drive	IB277:P – IB282:P	QB277:P – QB282:P
	PD for the 2nd drive	IB283:P – IB288:P	QB283:P – QB288:P
HVAC FC102	PROFIdrive standard telegram 1	IB289:P – IB292:P	QB289:P – QB292:P
	PCV	IB293:P – IB300:P	QB293:P – QB300:P
	PCD	IB301:P – IB320:P	QB301:P – QB320:P
DP/PA coupler	Current	IB321:P – IB325:P	N/A
	Voltage	IB326:P – IB330:P	N/A
Level transmitter	Main process value	IB331:P – IB335:P	N/A
	2nd cyclic value	IB336:P – IB340:P	N/A
	Display value	N/A	QB341:P – QB345:P
Pressure transmitter	Main process value	IB346:P – IB350:P	N/A
	2nd cyclic value	IB351:P – IB355:P	N/A
	3rd cyclic value	IB356:P – IB360:P	N/A
	Display value	N/A	QB361:P – QB365:P

Table 17: Consistent peripheral I/O address for slave modules in the sample code

All the read and write online sample code were done with the generic **Move** block instead of some sophisticated communication function block in order to reduce the complexity of the code, so that students can easily follow the sample code.

For each device, there is a diagnostic section which provides information for troubleshooting such as LED behaviours and error code.

Project Diary

A project diary was also created which clearly records all the significant achivements and events during the project. The project diary is weekly based and will also help future students to track down what has been done in the past as well as the problem encountered. It can combined with the Gantt chart provided to form a time management guide line for the future project developer for project planning.

(Refer to Appendix I Project Diary for the contents of the diary)

(Refer to Appendix II Project Gantt Chart for the contents of the Gantt chart)

CHAPTER 9: NETWORK DESIGN

Profibus DP Network

The Profibus network design in the ICSE laboratory was carried out to replace the old design. The new network design certainly has more flexibility and is easier to adapt to common requirements. **Figure 110** shows the new structure of the network.

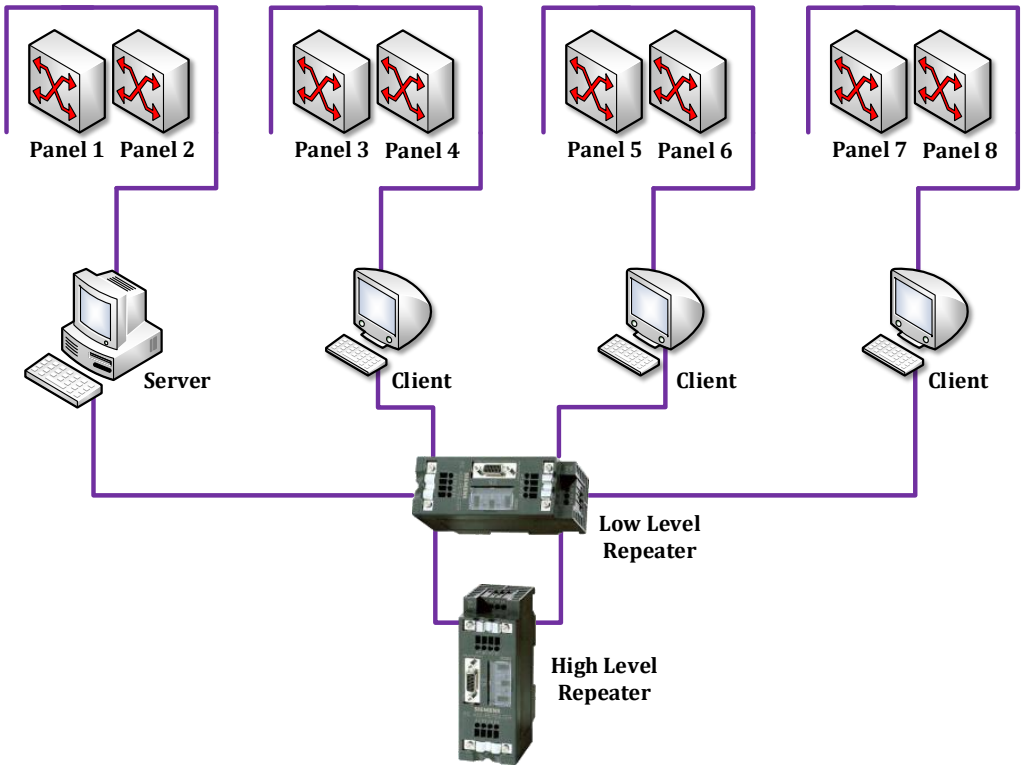


Figure 110: Designed Profibus DP network in ICSE laboratory

All the termination switches on both repeaters need to be switched on if no cross panel network exists. Each computer can be set up as a single WinCC station to communicate with devices on 2 panels. When there is a need to connect the devices on panels 1 & 2 and panels 3 & 4, the corresponding termination switch on the low level repeater must be switched off. The same principle applies to panels 5 & 6 and panels 7 & 8. This low level repeater is connected to the high level repeater with 2 cables coming from the first 4 panels and the second 4 panels respectively. When a need arises to join all 8 panels together, the termination switch on the high level repeater needs to be switched off with the first computer becomes the WinCC server and all the rest become the WinCC clients, if more than one computer is integrated within the network. The repeaters do not need to be powered if signal amplification is not required. [26]

Profibus PA Network

The Profibus PA network in the ICE laboratory was tested and confirmed including the inspection of the cables on the roof and the removal of the unused cables. **Figure 111** shows the structure of the network.

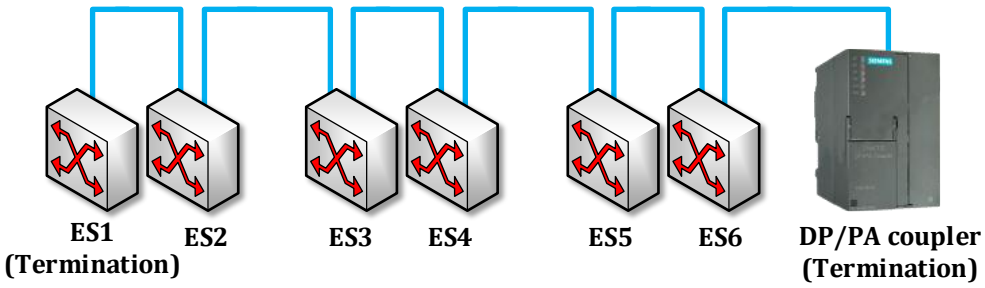


Figure 111: Profibus PA network in ICE laboratory

Each I/O panel has two M12 female connectors ready for the devices to connect. The termination was set on both ES1 I/O panel and DP/PA coupler as end points. Behind each I/O panel, there is a terminal block which is responsible for distributing connection for two M12 connectors on the I/O panel and also passing one cable to the next panel to loop through the signal.

CHAPTER 10: FUTURE RECOMMENDATIONS

WinCC Server and Client Communication

The WinCC server and client communication needs to be accomplished and the final outcome is to have client computers control the process via the server computer project. It must be confirmed whether the WinCC server computer needs to have server OS installed. In addition, the Web navigator component can also be integrated so that other computers within the network can access the process screen of WinCC on the internet explorer over Ethernet. [88]

Upgrade to WinCC Runtime Professional

The current TIA Portal license is WinCC Comfort and does not support a SCADA system. That is why WinCC explorer was used. However, the new released highest license of TIA Portal, known as WinCC Runtime Professional, has integrated the capability of SCADA to replace WinCC explorer. TIA Portal with WinCC Runtime Professional license will allow users to perform the whole configuration in one single software and achieve real TIA. [38]

HMI Touch Panel Ethernet Communication

It is urgent to find out how multiple HMI touch panels (maximum 4 in a single network) can be configured to control a single PLC over Ethernet to achieve real remote operator control. [61] [65] [78]

Level Transmitter Tank Upgrade

The current water tank which the Endress level transmitter was inserted has a very small cross section area and can be filled up very fast. This introduces some difficulties to control the level in the tank because the system is too sensitive and the significant communication delay. Hence the tank needs to be replaced with a large cross section area tank to compensate the delay reaction time.

Controller Design

It was proved that the number of the shared variable read from LabVIEW OPC server has a significant impact on the communication, causing a delay that generic PID controller could not handle. Thus, if this sort of control system needs to be implemented in the future, a more advanced controller such as DMC (Dynamic Matrix Control) will need to be designed and implemented to compensate for the huge delay.

PROFIdrive Profile

Siemens may release the instruction function block which is equivalent to FB36 in STEP 7 very soon. Once this instruction function block is available, the parameters of the both the SEW and Danfoss drives can be directly accessed via the PROFIdrive parameter channel. [9] [11] [16] [54]

Safety Operations

The safe operation feature needs to be integrated into the control system. For example, when the communication of the drive has lost, an interlock on the drive will be activated and also notify the master controller. This can be achieved from the drive parameters, but this function should be expanded to other devices. Therefore, a comprehensive solution is required.

CHAPTER 11: CONCLUSION

Through this project, all the major objectives were accomplished with lots of small tasks. Progress was made every day with a lot of help from the project supervisor and the associated technical officers such as profile policies and changing fittings. A project diary was written which contains all the individual events during the project on a weekly basis.

All the Profibus interface devices were fully functioning with their capabilities having been fully discovered. A good understanding and practise of the fieldbus communication was obtained after enormous telegram data exchanges.

The WinCC SCADA system has successfully communicated with multiple systems within the network and operated as expected. The miscellaneous configuration methods were used to take advantages of the flexible SCADA environment.

The Profibus DP network in the ICSE lab was set up to satisfy the demands of education. The Profibus PA network in the ICE lab was also tested and its connections were confirmed. Hence, at the time of writing, both networks in the laboratory are working so students can benefit from the outcome of this project.

The configuration instructions for devices and software were created during the configuration to ensure no important information is missing so that future students can easily follow the instructions to meet their own goals.

In conclusion, this project is very intensive and requires a lot of troubleshooting skills. However, it is a good opportunity to work with the actual system and gain some valuable experience for the future.

CHAPTER 12: REFERENCE

[1] Acromag, "Introduction to Profibus DP," *38 Pages,* 2002. (General information about Profibus DP includes GSD, C1/C2 master and reaction time).

[2] National Instruments, "Profibus Overview," *4 Pages,* 2010. (Information on DP V0/V1/V2 and FAQ).

[3] PNO, "Profibus Installation Guideline for Cabling and Assembly," *132 Pages,* 2006. (Information on wiring and interference of Profibus cable).

[4] PNO, "Profibus Installation Guideline for Planning," *230 Pages,* 2009. (Information on Profibus cabling and connectors).

[5] PNO, "Profibus Technical Description," *38 Pages,* 1999. (Information on bus cycle, cyclic and acyclic communications).

[6] Profibus Competency Center, "Profibus RS485 Wiring," *3 Pages,* 2009. (Information on Profibus wiring and different types of connectors).

[7] Siemens, "SIMATIC NET Profibus Network Manual," *350 Pages,* 2009. (Information on topologies, active/passive components and connector specifications).

[8] Siemens, "SIMATIC NET Profibus Networks Manual," *490 Pages,* 2000. (Information on Profibus network peripheral specifications).

[9] Siemens, "Drive ES SIMATIC V5.4 Function Block Library DRVDPS7," *162 Pages,* 2006. (Information on DS 47 for PROFIdrive parameter channel).

[10] PNO, "PROFIdrive System Description," *24 Pages,* 2011. (Information on PROFIdrive models, profile and mapping).

[11] Siemens, "SINAMICS S120 Function Manual," *982 Pages,* 2013. (Information on PROFIdrive parameter channel and data block 47).

[12] Siemens, "SINAMICS S120 Commissioning Manual," *412 Pages,* 2006. (Information on PROFIdrive telegram and process data).

[13] PNO, "PROFIsafe System Description," *26 Pages,* 2010. (Information on PROFIsafe specification).

[14] Phoenix Contact, "Profinet Basics," *34 Pages,* 2011. (Information on Profinet Specification).

[15] Siemens, "Profibus Cards," *6 Pages,* 1999. (Information on CP5611 PCI card capabilities).

[16] Siemens, "TIA Portal and STEP 7 Library Cross Reference List," 2011. (Information on instruction library cross reference list between TIA Portal and STEP 7).

[17] H. Xu, K. Gumireddy and Y. Zhang, "Profibus PA," Murdoch University, Perth, 2012. (Information on Profibus specifications).

[18] H. Xu, J. Neylan and J. Batson, "IC facility," Murdoch University, Perth, 2013. (Information on Profibus network and OPC server/client).

[19] C. Moss, J. Baker, F. Worku and S. Tyler, "IC facility," Murdoch University, Perth, 2013. (Information on the Profiubs PA bus termination).

[20] S. Tyler, C. Moss, J. Baker and F. Worku, "ICSE Profibus network," Murdoch University, Perth, 2013. (Information on HMI device compatibility).

[21] N. B. M. Nain, A. Tokhmechi and O. Bensaoud, "Profibus DP," Murdoch University, Perth, 2013. (Information on PLC MS communication).

[22] L. Morrison, T. Gittos, N. Kilburn and C. Woodard, "ICSE laboratory Profibus network," Murdoch University, Perth, 2013. (Information on MOVITRAC B modules).

[23] J. Vandepeer, W. Justin and M. Oeding, "ICSE laboratory Profibus network," Murdoch University, Perth, 2013. (Information on Moeller communication and VSD control/status word).

[24] B. Walker, "Development of a Profibus network and WinCC SCADA environment for educational purpose," Murdoch University, Perth, 2011. (Information on network configuration).

[25] B. Walker, "P&WCC A31 Laboratory Guide 1-6," Murdoch University, Perth, 2011. (Information on configurations includes PLC MS communication, MOVITRAC 07, FC102, WinCC flexible and WinCC).

[26] J. Martens, "Siemens Profibus system for teaching purposes," Murdoch University, Perth, 2012. (Information on ICSE Profibus network design and termination box).

[27] J. Martens, "You and your VSD tutorial 1-2," Murdoch University, Perth, 2013. (Information on MotionStudio control and SBus profile).

[28] R. Imber, "Siemens WinCC flexible tutorial set," *21 parts,* 2011. (Information on WinCC flexible operations include communication, objects, functions and script).

[29] B. Z. Liang, "Siemens WinCC application and development tutorial set," *13 Episodes,* 2011. (Information on WinCC operations include communication, objects, function and script).

[30] Siemens, "SIMATIC S7-300 Programmable Controller CPU Specifications, CPUs 312C to 314-2DP/PtP Reference Manual," *154 Pages,* 2001. (Information on technical data of PLC models).

[31] Siemens, "SIMATIC S7-300 CPUs and Modules," *124 Pages,* 2003. (Information on PLC modules).

[32] Siemens, "SIMATIC Communication with SIMATIC System Manual," *220 Pages,* 2006. (General information on different fieldbus communications with profiles).

[33] Siemens, "SIMATIC Standard Software for S7 and M7 Step 7 User Manual," *360 Pages,* 1997. (Information on Programming languages in Step 7).

[34] Siemens, "SIMATIC Software for S7-300 and S7-400 Standard Function Reference Manual Part 2," *106 Pages,* 2000. (Information on system FC and FB).

[35] Siemens, "SIMATIC System Software for S7-300/400 System and Standard Functions Reference Manual Vol.2," *756 Pages,* 2006. (Information on organization blocks).

[36] Siemens, "SIMATIC S7-SCL V5.3 for S7-300-400 Manual," *394 Pages,* 2005. (Information on SCL programming structure).

[37] Siemens, "SIMATIC S7-GRAPH V5.2 for S7-300/400 Programming Sequential Control System Manual," *232 Pages,* 2002. (Information on GRAPH actions).

[38] Siemens, "STEP 7 Professional V12.0," *9504 Pages,* 2013. (Information about everything in TIA Portal includes background and configuration, functions and devices).

[39] H. Berger, "Automating with STEP7 in STL and SCL," *530 Pages,* 2007. (Information on SCL addressing and conversion).

[40] Siemens, "SIMATIC RS485 Repeater Manual," *46 Pages,* 2011. (Information on repeater wiring and specifications).

[41] Eaton, "Easy500/700 User Manual," *304 Pages,* 2004. (Information on Moeller operations includes installation, functions, commissioning and diagnosis).

[42] Eaton, "Easy204-DP Operating Manual," *260 Pages,* 2004. (Information on Moeller telegram data exchange includes input/output and control level).

[43] SEW, "MOVITRAC B System Manual," *314 Pages,* 2011. (General information includes installation, specification, parameters and operations).

[44] SEW, "MOVITOOLS MotionStudio Manual," *160 Pages,* 2009. (Information on MotionStudio communication, configuration and commissioning).

[45] SEW, "Fieldbus Interface DFP21B Profibus DP-V1 Manual," *114 Pages,* 2006. (Information on Profibus communication includes wiring, parameter channel and process data).

[46] SEW, "MOVITRAC 07 Communication Manual," *60 Pages,* 2003. (Information on SBus and MOVILINK profile).

[47] SEW, "MOVIDRIVE MDX60B/61B Communication and Fieldbus Unit Profile," *144 Pages,* 2009. (Information on the contents of control word and status word).

[48] SEW, "MOVIDRIVE Serial Communication Manual," *92 Pages,* 2001. (Information on parameter channel data).

[49] Danfoss, "VLT HVAC Drive Instruction Manual," *172 Pages,* 2007. (Information on installation of FC100 and testing).

[50] Danfoss, "VLT MCT 10 Set-up Software Manual," *139 Pages,* 2012. (Information about communication, drive configuration and commissioning in MCT 10).

[51] Danfoss, "VLT HVAC Drive Programming Guide," *343 Pages,* 2007. (Information about all the drive parameters with descriptions).

[52] Danfoss, "VLT Profibus DP MCA101 Brochure," *2 Pages,* 2012. (Information on MCA101 option includes its capabilities).

[53] Danfoss, "Profibus Operating Instruction," *101 Pages,* 2006. (Inforamtion on FC100 data exchange using MCA101 includes sample telegram examples).

[54] Danfoss, "Profibus DP V1 Design Guide," *40 Pages,* 2002. (Information on PROFIdrive parameter channel in STEP 7 includes sample S7 program).

[55] Danfoss, "VLT Option Overview," *12 Pages,* 2008. (Information about drive options and functionalities).

[56] Danfoss, "VLT HVAC Basic Drive Design Guide," *93 Pages,* 2011. (General inforamtion about FC100).

[57] Siemens, "SIMATIC HMI Device Operating Instruction," *380 Pages,* 2008. (General information on installation, communication, functions and operations).

[58] Siemens, "How can you transfer a WinCC (TIA Portal) configuration to an operator panel using MPI/Profibus?," *26 Pages,* 2012. (Information on HMI device configuration and communication).

[59] Siemens, "SIMATIC HMI WinCC flexible 2008 User Manual," *466 Pages,* 2008. (General information includes communication, configuration, objects and commissioning).

[60] Siemens, "SIMATIC HMI WinCC flexible 2008 Introduction to WinCC flexible," *32 Pages,* 2010. (Information on HMI device function dependency).

[61] Siemens, "SIMATIC HMI WinCC flexible 2008 Basic Requirments," *42 Pages,* 2010. (Information on communication includes HMI connection and area pointers connection).

[62] Siemens, "SIMATIC HMI WinCC flexible 2008 SIMATIC S7," *82 Pages,* 2010. (Information on Profibus configuration parameter and area pointer word).

[63] Siemens, "SIMATIC HMI WinCC flexible 2008 Transfer," *46 Pages,* 2010. (Information on different transfer modes with diagnosis).

[64] Siemens, "SIMATIC HMI WinCC flexible 2008 Getting Started - First Time User," *81 Pages,* 2008. (Simple object configuration for entry level user).

[65] Siemens, "SIMATIC HMI WinCC flexible 2008 Getting Started - Options," *123 Pages,* 2008. (Inforamtion on simple configuration of Sm@rtAccess, Sm@rtService and OPC).

[66] Siemens, "SIMATIC HMI WinCC flexible 2008 Getting Started - Power User," *175 Pages,* 2008. (Information on advanced configuration of recipe, faceplate and report).

[67] Siemens, "SIMATIC HMI WinCC flexible 2008 Engineering System," *60 Pages,* 2010. (Description of WinCC flexible configuration environment with function, graph, ext and object lists).

[68] Siemens, "SIMATIC HMI WinCC flexible 2008 Display and Operating Objects," *88 Pages,* 2009. (Brief description of all the objects, elements and controls).

[69] Siemens, "SIMATIC HMI WinCC flexible 2008 Creating Screens," *188 Pages,* 2010. (Information on screen navigation, animation and faceplate).

[70] Siemens, "SIMATIC HMI WinCC flexible 2008 Operation in Runtime," *122 Pages,* 2010. (Information on general operation of objects, controls and views).

[71] Siemens, "SIMATIC HMI WinCC flexible 2008 Simulating Tags," *24 Pages,* 2010. (Information on the operation of Runtime simulator).

[72] Siemens, "SIMATIC HMI WinCC flexible 2008 User Administration," *62 Pages,* 2010. (Information on user group authorization and user view setup).

[73] Siemens, "SIMATIC HMI WinCC flexible 2008 Working with Alarms," *82 Pages,* 2010. (Information on alarm configuration includes alarm class, types and groups).

[74] Siemens, "SIMATIC HMI WinCC flexible 2008 Working with Tags," *68 Pages,* 2010. (Information on tag properties, types and connections).

[75] Siemens, "SIMATIC HMI WinCC flexible 2008 Working with Projects," *114 Pages,* 2010. (Information on WinCC project setting, cross reference and troubleshooting).

[76] Siemens, "SIMATIC HMI WinCC flexible 2008 System Alarms," *46 Pages,* 2009. (Information on system alarm code and description).

[77] Siemens, "SIMATIC HMI WinCC flexible 2008 System Functions," *260 Pages,* 2009. (Information on function dependency and function meaning associated with events).

[78] Siemens, "SIMATIC HMI WinCC flexible 2008 HTTP Protocol," *24 Pages,* 2010. (Information on configuration of HTTP server and clients).

[79] Siemens, "SIMATIC DPPA Bus Coupler Manual," *158 Pages,* 2000. (Information on the GSD modules).

[80] Siemens, "SIMATIC Bus links DPPA Coupler, active field distributors, DPPA Link and Y link Operating Instruction," *266 Pages,* 2011. (Information on the structure of PA network and configuration).

[81] Endress+Hauser, "Levelflex M FMP40 Operating Instruction," *120 Pages,* 2006. (Information on communication modules and structure).

[82] Endress+Hauser, "Levelflex M FMP40 Technical Information," *72 Pages,* 2008. (Information on transmitter specification and operating design).

[83] Endress+Hauser, "Deltabar S FMD76/77/78, PMD70/75 Operating Instructions," *100 Pages,* 2004. (Information on communication modules and structure).

[84] Endress+Hauser, "Deltabar S PMD70/75, FMD76/77/78 Technical Information," *96 Pages,* 2010. (Information on transmitter specification and operating design).

[85] Siemens, "WinCC V6 Communication Manual," *550 Pages,* 2004. (Information on various types of communication includes individual project examples).

[86] Siemens, "WinCC Configuration Manual Vol. 1," *320 Pages,* 1999. (Inforamtion on object properties, C actions and symbol library).

[87] Siemens, "WinCC Configuration Manual Vol. 2," *456 Pages,* 1999. (Information on configuration of tags, screens, objects, controls and user archive).

[88] Siemens, "WinCC Configuration Manual Vol. 3," *242 Pages,* 1999. (Information on multi-client, server, redundancy and user archive).

[89] Siemens, "SIMATIC HMI WinCC System Overview," *88 Pages,* 2011. (Information on WinCC components and work flow).

[90] Siemens, "SIMATIC HMI WinCC V7.0 Working with Projects," *154 Pages,* 2011. (Information on WinCC project settings).

[91] Siemens, "SIMATIC HMI WinCC V7.0 Getting Started," *271 Pages,* 2011. (Information on simple configuration includes communication, screens and archive).

[92] Siemens, "SIMATIC HMI WinCC V7.0 SIMATIC S7 Protocol Suite," *102 Pages,* 2011. (Information on various types of communication drivers).

[93] Siemens, "SIMATIC HMI WinCC V7.0 Communication - Diagnostics," *72 Pages,* 2011. (Information on various types of diagnosis methods for connection and tag.).

[94] Siemens, "SIMATIC HMI WinCC V7.0 Working with Tags," *68 Pages,* 2011. (Information on structure tags and data types).

[95] Siemens, "SIMATIC HMI WinCC V7.0 Creating Process Pictures," *776 Pages,* 2011. (Information on properties of all the objects and graphics designer).

[96] Siemens, "SIMATIC HMI WinCC V7.0 Process Communication," *42 Pages,* 2011. (Information on data format adaptation).

[97] Siemens, "SIMATIC HMI WinCC V7.0 Working with Controls," *74 Pages,* 2011. (Information on configuration and parameters of control elements).

[98] Siemens, "SIMATIC HMI WinCC V7.0 Process Value in the Form of Trends in Process Pictures," *76 Pages,* 2011. (Information on OnlineTrendControl configuration).

[99] Siemens, "SIMATIC HMI WinCC V7.0 Process Value Output in Table Format," *48 Page,* 2011. (Information on OnlineTableControl configuration).

[100] Siemens, "SIMATIC HMI WinCC V7.0 Setting up a Message System," *123 Pages,* 2011. (Information on alarm message configuration).

[101] Siemens, "SIMATIC HMI WinCC V7.0 Display of Message during Runtime," *76 Pages,* 2011. (Information on AlarmControl configuration).

[102] Siemens, "SIMATIC HMI WinCC V7.0 Process Value Output as a Function of Another Tag," *58 Pages,* 2011. (Information on FunctionTrendControl configuration).

[103] Siemens, "SIMATIC HMI WinCC V7.0 Message Archiving," *30 Pages,* 2011. (Information on directly accessing data from WinCC database).

[104] Siemens, "SIMATIC HMI WinCC V7.0 Archiving Process Value," *102 Pages,* 2011. (Information on different types of archiving).

[105] Siemens, "SIMATIC HMI WinCC V7.0 Structure of the User Administration," *52 Pages,* 2011. (Information on automatic log off function).

[106] Siemens, "SIMATIC HMI WinCC V7.0 OPC Channel," *48 Pages,* 2011. (Information on OPC DA/XML/UA and OPC item manager).

[107] Siemens, "SIMATIC HMI WinCC V7.0 OPC Open Connectivity," *104 Pages,* 2011. (Information on communication between WinCC and Excel via OPC).

[108] Siemens, "SIMATIC HMI WinCC V7.0 Horn," *64 Pages,* 2011. (Information on horn configuration).

[109] Siemens, "SIMATIC HMI WinCC V7.0 SmartTools," *132 Pages,* 2011. (Information on creating dynamic wizard).

[110] Siemens, "SIMATIC HMI WinCC V7.0 ANSI-C for Creating Functions and Actions," *95 Pages,* 2011. (Information on GSC diagnosis and Runtime).

[111] Siemens, "SIMATIC HMI WinCC V7.0 VBS for Creating Procedures and Actions," *152 Pages.,* 2011. (Information on VBS strcture and sample program).

[112] Siemens, "SIMATIC HMI WinCC V7.0 MDM - WinCC: Scripting (VBS, ANSI-C, VBA) System Manual," *2532 Pages.,* 2008. (Information on references for VBS, ANSI-C and VBA).

[113] OPC Foundation, "OPC Data Access Automation Specification V2.02," *100 Pages,* 1999. (Information on VBA syntax for DAAutomation 2.0).

[114] National Instruments, "Distributed Signal Conditioning and I/O Modules," *1 Page,* 2002. (Information on distributed I/O module capabilities).

Software Used

- Siemens STEP 7 V5.5
- Siemens TIA Portal V11
- Siemens WinCC V7.0
- National Instruments LabVIEW 2011
- Eaton EASYSOFT-Pro V6.91
- SEW MOVITOOLS MotionStudio V5.90
- Danfoss MCT 10 V3.21
- Microsoft Excel 2010
- Microsoft Word 2013
- Microsoft PowerPoint 2013
- Microsoft Access 2010
- Microsoft Visio 2013
- Microsoft Project 2013
- Microsoft SQL Server Management Studio Express 2003
- Adobe Photoshop CS5
- Adobe After Effects CS5
- Autodesk 3ds Max 2011
- Gedit V2.29
- NeoSpeech V2.0
- Easy GIF Animator V5.4

CHAPTER 13: APPENDICES

Appendix I Project Diary

This section contains the contents of the diary which clearly records the achievements made during the project.

Date	Achievements
Week -6 (17th Jun – 23rd Jun)	• Getting familiar with TIA Portal. • Revision of PLC functions.
Week -5 (24th Jun – 30th Jun)	• MPI & Profibus communication. • Profibus termination inspection. • Project panel setup. • Configuring PLC MS Profibus communication. • PLC MS data exchange.
Week -4 (1st Jul – 7th Jul)	• Integrating EASY719 into the Profibus network. • Finding the difference between R/S and I/Q.
Week -3 (8th Jul – 14th Jul)	• Using input/output level modules for data exchange. • Using control level modules for data exchange.
Week -2 (15th Jul – 21st Jul)	• MOVITRAC B keypad control. • Getting familiar with parameters.
Week -1 (22nd Jul – 28th Jul)	• Using MotionStudio to get familiar with MOVITRAC B. • Testing download and upload to MOVITRAC B from MotionStudio. • Terminal control. • Configuring gateway in MotionStudio.
Week 1 (29th Jul – 4th Aug)	• Startup and scope in MotionStudio. • 1 gateway with 2 MOVITRAC B communication. • Gateway Profibus communication. • Looking for information about control and status word.
Week 2 (5th Aug – 11th Aug)	• Process data exchange. • Fixing the overwritten process data problem. • Set up a computer in Mechatronic room.
Week 3 (12th Aug – 18th Aug)	• Plan changed due to isolation switch fault. • Configured HMI in TIA Portal.
Week 4 (19th Aug – 25th Aug)	• HMI screen design in TIA Portal. • Configuring HMI in WinCC flexible. • HMI screen design in WinCC flexible.
Week 5 (26th Aug – 1st Sep)	• Installing MCT 10 for FC102 configuration. • Getting familiar with parameters. • Startup motor using LCP.
Week 6 (2nd Sep – 8th Sep)	• HVAC FC102 terminal control. • Ordering a Danfoss Profibus adaptor Sub-D9 connector. • Using MCT 10 to monitor and change parameter values. • Investigating PROFIdrive profile. • PCD data exchange. • PCV data exchange.
Week 7 (9th Sep – 15th Sep)	• Installing Profibus adaptor Sub-D9 connector. • Setting up WinCC OPC server/client. • Standard object configuration and testing in WinCC.
Week 8 (16th Sep – 22nd Sep)	• Tag logging and alarm logging. • Configuring DP/PA coupler and Profibus interface transmitters.
Week 9 (23rd Sep – 29th Sep)	• Consolidating SCL programming language. • Changing pipe fitting of the level transmitter tank. • WinCC MPI network communication. • 3D modelling.

Week 10 (30th Sep – 6th Oct)	• Design control system. • Investigate miscellaneous functions in WinCC.
Week 11 (7th Oct – 13th Oct)	• Testing the Profibus interface transmitters. • Proving the effect of termination and repeater. • Choosing OPC communication scheme. • Writing SCL program for data exchange. • Testing the control system in LabVIEW.
Week 12 (14th Oct – 20th Oct)	• Prepare symbol table and tags in TIA Portal and WinCC. • Testing ANSI-C and VBScript code in WinCC. • Developing VBA program to access WinCC tags.
Week 13 (21st Oct – 27th Oct)	• HMI Touch panel screen design. • TIA Portal program testing. • WinCC tag logging and alarm logging configuration.
Week 14 (28th Oct – 3rd Nov)	• Render HMI graphics. • Testing control system.
Week 15 (4th Nov – 10th Nov)	• Laboratory Profibus network setup • Project demonstration. • Preparing presentation slides.
Week 16 (11th Nov – 17th Nov)	• Final presentation. • Organizing documentations.

Week -6 (17th Jun – 23rd Jun)

Getting Familiar with TIA Portal

Since TIA Portal is a new configuration software and pretty much all the previous documentations are based on SIMATIC manager, it is extremely important to become familiar to it as soon as possible. Even though TIA Portal is just a substitution of SIMATIC manager, the user interface, location of functions, naming conventions and workflow are not necessarily identical. Moreover, TIA Portal will be the only Profibus network configuration tool and therefore, spend some time to get familiar with the TIA Portal environment can be considered as the very first step of this project.

Revision of PLC Functions

The next task after learning TIA Portal environment got to be the revision of PLC programmin. Since Profibus communication would require a lot of data exchange, a comprehensive understanding of differnet data types and lengthes are quite important in this case. As the future goal of this project is to design and implement a control system using Profibus communication, a lot of function calls and data blocks would be very handy to simplify the program. Due to the complexity of the control system, the high level programming language such as SCL and GRAPH will be mainly used rather than ladder logic diagrams.

Week -5 (24th Jun – 30th Jun)

MPI & Profibus Communication

The MPI and Profibus communications were successfully tested with a USB interface PC/PPI cable and a RS485 Profibus cable with Sub-D9 pin connector respectively. At this stage, the communication was only between the PLC and the PC.

Profibus Termination Inspection

As the wiring in those Profibus connectors in the laboratory were not consistent, the termination switches might not work in the expected way. For those connectors with termination switches, the termination is supposed to stop the data from passing to the next device, and there are 2 Profibus cable slots in the connector to provide a daisy chain network. One slot is to take the data from the previous device and another one is to pass the data to the next device. It was found that some cables in the connectors had the reversed order which caused the termination problem. There was another kind of Profibus connector which fix the wire by clamping the cable, but a lot of cable found in this type of connectors were not stripped at all which would not be able to conduct current. In order to avoid problems caused by the connector in the future, all the cables found in the laboratory were removed from the connectors to confirm the connection. After the inspection, all the cables were re-plugged into the connectors and labelled properly.

Project Panel Setup

This project required a number of testings and this could be quite tedious because the project developer needed to walk between the PC and the devices on the laboratory panel to see the changes made. One of the solutions to this was to mount all the required devices on a portable project panel. All the necessary devices were then mounted on a project panel which includes 2 PLCs, 1 EASY719 controller, 1 HMI, 2 MOVITRAC B with 1 gateway and a FC102. A few external switches, potentiometers and LEDs were also mounted on the panel for testing purpose.

Configured PLC MS Profibus Communication

The PLC master-slave Profibus communication was established in TIA Portal by introducing an I-slave transfer area for data exchange and creating OB100 and OB82 in both PLCs. Even though 2 PLCs had the master-slave relationship, it was required to download to each PLC individually (download to slave first then master). After the configuration has been downloaded into the PLCs, both PLCs need to be switched to STOP mode and switched back to RUN mode to get rid of the fault. A few tests were carried out to confirm that the order to switch them is not important.

PLC MS Data Exchange

The master-slave data exchange was implemented without any issues. Both master and slave PLCs were able to read and write to each other using the I-slave transfer area.

Week -4 (1st Jul – 7th Jul)

Integrate EASY719 into the Profibus Network

The EASY204-DP GSD file was successfully installed in TIA Portal and the station was dragged into the network and linked to the Profibus subnet. However, EASY204-DP could not be integrated into the network after downloading. It was found out that at least one of the module of the EASY204-DP needs to be dragged onto the rack. Further investigation showed that, certain modules could not exist on the rack at the same time, otherwise, PLC would indicate a bus fault. Another issue was that the Profibus address of EASY204-DP can only be set from EASY719 controller, which means that assign a random Profibus address in TIA Portal and download it to EASY204-DP would not work.

Found out the Difference between R/S and I/Q

All previous documentation mentioned about R inputs and S outputs, but the actual input and output in EASY719 are I inputs and Q outputs which were quite confusing. Initially, R and S were thought as equivalent to I and Q, but further investigation showed that the number of them are different, which proved that they must represent different functions. Finally, by reading the manual extensively, a general understanding was obtained to point out that R and S are used as inputs and outputs for expansion unit only. This finding then proved that all the work done by previous students were in a wrong direction because even though R and Q could somehow link to I and Q by writing some program in EASY719, this is not the designed way to perform data exchange.

Week -3 (8th Jul – 14th Jul)

Using Input/Output Level Modules for Data Exchange

The input/output level modules were used to read and write to expansion unit inputs and outputs. Both 1 byte and 3 bytes input/output modules were tested and some positive outcomes were obtained. Some code in were developed in Moeller to link R/S to I/Q to indirectly read the actual input and control the actual output even it was not the designated way.

Using Control Level Module for Data Exchange

The control module is the most comprehensive module for data exchange between PLC and EASY719. All the available telegrams were tried out to get state values as well as parameters from different functions. The official user instruction contains a lot of critical errors, so sometimes need to try some random telegrams to find the correct one.

Week -2 (15th Jul – 21st Jul)

MOVITRAC B Keypad Control

By reading the manual, general idea of operation was obtained. However, MOVITRAC B could not be enabled from the keypad for some reason. It was quickly found out that MOVITRAC B needs to enter FBG manual control mode to be controlled via the keypad. After that, the keypad module was able to start/stop the drive as well as control the speed of the motor via the potentiometer.

Getting Familiar with Parameters

A lot of effort were putting in to get familiar with all the parameters and picking parameters which are more relevant to the project's perspective. A few different versions of manuals were combined to give a comprehensive understanding.

Week -1 (22nd Jul – 28th Jul)

Using MotionStudio to Get Familiar with MOVITRAC B

Since MOVITRAC B contains a huge range of parameters which would take long time to find out what each one does, MotionStudio was used to help to shorten this period. A USB11A module was used for communication and MOVITRAC B was detected after a network scan.

Testing Download and Upload to MOVITRAC B from MotionStudio

Some parameter were changed and downloaded to MOVITRAC B, but the online parameter stayed the same. The reason for that was quickly addressed as the drive parameter would not change due to a download while in online mode. Another issue was that some numeric parameters could not be changed as they supposed to, but they can be changed on the MOVITRAC B unit.

Terminal Control

The digital inputs on MOVITRAC B were supposed to control the MOVITRAC B. Some simple operations were assigned to the digital inputs, but they did not seem to work. This problem had been solved and the reason was that the terminal control cannot be activated while in FBG manual control because FBG manual control has higher priority.

Configuring Gateway in MotionStudio

The gateway was connected to MOVITRAC B using a Profibus cable according to the diagram in the manual. However, the network scan in MotionStudio can only detect gateway. It was found out that ***P880 SBus protocol*** needs to be changed to Movilink, and this solved the problem. A network scan was then performed and was able to detect MOVITRAC B under the gateway.

Week 1 (29th Jul – 4th Aug)

Startup and Scope in MotionStudio

Startup in MotionStudio was used to control MOVITRAC B and it worked quite well. From the manual, Scope in MotionStudio is able to plot the response of MOVITRAC B. The plot only showed a straight line no matter how the speed has been changed and the reason for this was then found out to be the short sample time.

1 Gateway with 2 MOVITRAC B Communication

Again, a Profibus cable was connected between the first and the second MOVITRAC B to create a daisy chain network. The gateway, however, was only able to detect one of the MOVITRAC B. Making sure that all the devices have the same baud rate and unique SBus address also does not help. The Autosetup DIP switch on the gateway was accidentally reset and this solved the issue.

Gateway Profibus Communication

Gateway was successfully integrated into the Profibus network, but the **Parameter channel** slave module could not read and write to MOVITRAC B. The return code indicated as illegal parameter index according to the manual. It was realized that the **Parameter channel** should not take data from MOVITRAC B, because there is no index in the telegram indicate the address or number of the MOVITRAC in case of multiple MOVITRAC B in the network. The index for gateway parameters was tested and it worked as expected.

Looking for Information About Control and Status Word

The structure and the format of control and status word were not found in any version of gateway or MOVITRAC B manuals. However, some important information about process data were found in MOVITRAC 07 manual. As MOVITRAC 07 has significantly less capabilities, it could not be used to deduce the structure of process data for MOVITRAC B. Finally, a MOVIDRIVE manual was found and was considered as a suitable material to interpret MOVITRAC B process data structure.

Week 2 (5th Aug – 11th Aug)

Process Data Exchange

The PLC was only able to detect the first MOVITRAC B connected into the Profibus network. Different SBus address, group addresses, and protocols were changed to fix this problem, but nothing worked. The data for the second MOVITRAC B could be seen from the Bus diagnostics from ***P094*** to ***P099***, but not from the gateway point of view. This was quite strange because the data displayed in MotionStudio was only from the USB11A module connected between PC and the gateway which meant that the gateway was passing the data to the second MOVITRAC B, but it could not recognize MOVITRAC B. Luckily, the last idea, which was to reset the Autosetup DIP switch worked.

Fixed the Overwritten Process Data Problem

While writing process data to two MOVITRAC B, the PI data could be read without any issues, but the PO data written to the first MOVITRAC B was kept overwriting to the second MOVITRAC B. This issues could not be addressed until the factory setting was loaded from ***P802***.

Set up a Computer in Mechatronic Room

As the new semester will start shortly, the students would occupy the computers and panels in the laboratory quite often. a computer in the Mechatronic room was set up for the life time of the project.

Week 3 (12th Aug – 18th Aug)

Plan Changed due to Isolation Switch Fault

The initial plan was to configure HVAC FC102 before HMI touch panel, but HVAC FC102 tripped the 3 phase power while switching off the power supply in the Mechatronic room. The isolation switch in the switch box was causing this problem, therefore, the decision was made to configure HMI touch panel first which would leave some time for maintenance people to fix the isolation switch.

Configured HMI in TIA Portal

The configuration of HMI was unsuccessful because TP 177B 6" was configured as a master in the network. After a bit reading of manuals and research, TP 177B 6" was integrated into the Profibus network without any errors. Once launched the transfer to TP 177B 6" from TIA Portal, it indicated that the OS version on TP 177B 6" was too old and needs to be upgraded. After the upgrade finished, the program was successfully loaded to TP 177B 6".

Week 4 (19th Aug – 25th Aug)

HMI Screen Design in TIA Portal

Different elements and controls were created on the HMI screen and tested. The software configuration was pretty straight forward with the friendly user interface. The Runtime simulation was not able to run because of the policy problem, and this problem also blocked the communication between WinCC flexible and TP 177B 6".

Configured HMI in WinCC flexible

There was a computer in the instrumentation laboratory named EEPROJ-01 which had higher privilege to use Runtime simulation in TIA Portal and configure TP 177B 6" using WinCC flexible. Even though TIA Portal worked well as a substitution to configure HMI touch panel, it was still worth to use WinCC flexible to configure HMI touch panel and also compare the capability of two configuration software. On the EEPROJ-01 computer, without the usual policy limitation, the communication between WinCC flexible and HMI was established. However, it required to load a different HMI OS version other than the one got from TIA Portal. In other words, the HMI OS versions required for TIA Portal and WinCC flexible are different and reload of OS is required while changing configuration software.

HMI Screen Design in WinCC flexible

The HMI configuration in WinCC flexible was extremely easy after accomplishing it in TIA Portal because nearly all the functionality are the same. The problem was only to locate the same function in WinCC flexible.

Week 5 (26th Aug – 1st Sep)

Installing MCT 10 for FC102 Configuration

Even though the maintenance people had not fixed the 3 phase isolation switch tripping issue, the confguration of HVAC FC102 must start. HVAC FC102 contains much more parameters than that of MOVITRAC B which was shocking. By searching on the Danfoss website, it provided a software called MCT 10 for diagnosis. For safety reason, MCT 10 must be installed and used on a laptop to avoid damage to the USB port.

Getting Familiar with Parameters

A lot of effort were put into getting general understanding of all the parameters by combining the descriptions in MCT 10, LCP and user manual. It took 2 days to just write all the parameters with their assignments down.

Startup Motor Using LCP

The motor could not start up after following the instructions from the manual. It probably because there were some sorts of interlocks blocked the running function or some parameters with certain assignments that has higher priority to disable the motor. All the parameters were restored to the default factory settings and then the motor started to work and the speed could be adjusted from the LCP.

Week 6 (2nd Sep – 8th Sep)

HVAC FC102 Terminal Control

Digital inputs were used to control the behaviour of FC102 and changing the speed of the motor via an external potentiometer. Relay, pulse and analog output functions were investigated and confirmed as well.

Order a Danfoss Profibus Adaptor Sub-D9 Connector

The MCA101 Profibus card only came with some terminals which let the Profibus cable to wire in and did not have a standard Sub-D9 connector for the Profibus connector to plug in. This introduced some inconsistency in the network and the LCP102 module must be taken off to switch the bus termination on/off. A Danfoss Profibus adaptor Sub-D9 connector was found on a Danfoss brochure which was compatible with HVAC FC102 and MCA101. An order was made to Danfoss to deliver the connector to University.

Using MCT 10 to Monitor and Change Parameter Values

The changes were able to be made from the online parameter data in MCT 10 which is much easier compare to manually change the parameters from LCP102. A Scope folder was also created to monitor multiple channels in real time for diagnosis purpose.

Investigating PROFIdrive Profile

HVAC FC102 supports PROFIdrive profile which could be used to get values, attributes and data types from HVAC FC102 parameters. The PROFIdrive parameter channel esd linked to a data set 47 which is responsible for acyclic communication and there was a system function block called FB PDAT_AC2 which could do this job as a whole. However, this function block was not available in TIA Portal and therefore PROFIdrive parameter channel could not be used at this stage.

PCD Data Exchange

PPO Type 8 **Word consistent** module was used to test PCD communication. Both read and write PCD were tested and the outcomes were positive. The problem encountered was that the high speed limit value could not be modified. It was then found out that the value sent to the drive was too large so it held the old value.

PCV Data Exchange

The PCV communication was implemented to read/write to the drive parameters. The Sub-index of PCV telegram could not work properly and keep getting rejected from the drive. The causing of this issue was later on addressed by carefully checking the telegram sequence. The sub-index contained 2 bytes and the first byte is not used.

Week 7 (9th Sep – 15th Sep)

Installing Profibus Adaptor Sub-D9 Connector

The Profibus adaptor Sub-D9 connector from Danfoss had arrived as promised. The decoupling plate for mounting Profibus cable on the drive was taken off and was replaced by the new Profibus adaptor Sub-D9 connector. The connector was properly mounted on the drive and worked as expected.

Setting up WinCC OPC Server/Client

A WinCC OPC server was set up and LabVIEW was able to read and write to WinCC tags. Inversely, a LabVIEW OPC server was also set up and WinCC was also able to read and write to those shared variables as an OPC client.

Standard Object Configuration and Testing in WinCC

The normal operation of all the major objects such as buttons, tick box and slider were tested. The function of WinCC tag simulator and channel diagnosis tools were also explored.

Week 8 (16th Sep – 22nd Sep)

Tag Logging and Alarm Logging

Tag logging and alarm logging were completed with data and message archived in the database. Relative controls such as trend, table and alarm were also integrated into the screen.

Configuring DP/PA Coupler and Profibus Interface Transmitters

The DP/PA coupler was wired up and tested within the Profibus network with two Profibus interface transmitters. The configuration was successful with the internal PA termination resistor in the DP/PA coupler.

Week 9 (23rd Sep – 29th Sep)

Consolidating SCL Programming Language

Due to the complexity of the control system which will be designed, SCL text based language was considered as the major programming language for the PLC. The capability of SCL was further discovered such as scaling, splitting and masking. Some statement sample code from the manual were also collected as a reference for later use.

Changing Pipe Fitting of the Level Transmitter Tank

The tank with a Profibus interface level transmitter was equipped with a connector for a constant water input. Therefore, this tank was not suitable for any control system. A fitting suitable for a control valve was replaced so that the flow rate of the incoming water could be changed.

WinCC MPI Network Communication

One of the problems in the Profibus PA system on DP/PA coupler and transmitters was that the fixed data transmission speed is different to the default Profibus transmission speed. An idea was to create a secondar PLC master to collect all the data from the PA system and pass these data to the priamry PLC master. The research showed that TIA Portal does not support MPI global data communication anymore, so the secondary PLC master must communicate with WinCC directly via the communication driver.

3D Modeling

A series of 3D device models were started to build in 3D applications in order to give a better representation for demonstration and interface. The goal was to build a 3D model for each of the device which had been used in this project.

Week 10 (30th Sep – 6th Oct)

Design Control System

The control system had been designed with 3 control loops which involved level, temperature and pressure controls. The control system consisted of 2 parts. The first part was the controller part which collected all the data and performed all the calculations. The second part was the actual system which contained valves, pumps, heat exchanger, etc. TIA Portal was responsible for the configuration of the control system, whereas WinCC visualized and controlled the system and LabVIEW only passed the data through via OPC.

Investigate Miscellaneous Functions in WinCC

Further investigation on WinCC was done by reading huge amount of manuals and research. The purpose was to discover the wide range of WinCC capabilities. Functions such as **Menu and toolbars**, horn and some ActiveX were applied in WinCC.

Week 11 (7th Oct – 13th Oct)

Testing the Profibus Interface Transmitters

Both level and pressure transmitters were tested and the data obtained from the process were converted to the actual value using SCL in TIA Portal. The value did represent the physical process very well and they were ready to be integrated into the control system.

Proving the Effect of Termination and Repeater

The actual bus signal was displayed on an oscilloscope to see the effect of termination and the importance of the repeater. The connection without a termination demonstrated some amplification of noise and the connection with the repeater reduced the level of noise.

Choosing OPC Communication Scheme

As WinCC and LabVIEW would exchange data via OPC to implement the control system designed earlier, the OPC server and client need to be selected from them. Since OPC technology often introduces a significant delay, a few tests were carried out to find out the communication scheme that has the minimum delay. The outcome was that using LabVIEW OPC server would give minimum delay for both read and write access.

Writing SCL Program for Data Exchange

The SCL program was written to manipulate data such as scaling, converting and selecting within function blocks. The program had been tested without connecting to the actual devices.

Testing the Control System in LabVIEW

The control system designed including 3 control loops and needs to be physically implemented in the laboratory. As most of the panels only have 2 analog outputs available and some of the I/O modules were not working, a few testings were carried out to select a most suitable panel to link the system. The second panel was eventually decided to be used to implement the control system as it satisfied all the requirements.

Week 12 (14th Oct – 20th Oct)

Prepare Symbol Table and Tags in TIA Portal and WinCC

The symbol table was created in TIA Portal for all of the devices. All the relative tags were also added to the corresponding connection and group in WinCC. The data type and adapt format were also tested to ensure the value seen in TIA Portal and WinCC were identical. [96]

Testing ANSI-C and VBScript Code in WinCC

In order to achieve a flexible configuration, ANSI-C and VBScript code were decided to be used for the configuration only. As ANSI-C has more advantages on tag manipulation and VBScript is good at editing object properties, ANSI-C was focusing on the process tag value dynamics and VBScript was mainly used for animations. Some code were developed and tested on a small scale system.

Developing VBA Program to Access WinCC Tags

A VBA program was developed in Excel using an OPC add-on provided by WinCC to read and write WinCC tag values. The code is able to read value, access time and also the quality of the tag value. The program was then expanded to read and write multiple tags at the same time and all the WinCC tags were entered.

Week 13 (21st Oct – 27th Oct)

HMI Touch Panel Screen Design

A few screens were created in the HMI touch panel to allow operators to control and monitor the process remotely. The HMI touch panel was bound to the primary PLC and was only designed to include a few most important functions due to the limited physical screen size.

TIA Portal Program Testing

The testing of the PLC program was focused on the input and output of the slave devices and was done with the watching window. Basically, value was written modified in the watching window to the individual device FB and check whether the calculation gives the correct result and the program sends correct data to the device.

WinCC Tag Logging and Alarm Logging Configuration

Some of the important process values were selected for tag logging to reduce the workload of WinCC. The alarm was classified into 2 types, warning and failure. Each of them was linked to alarm word pre-built in the PLC. The alarm messages were also set to provide additional information about the alarms.

Week 14 (28th Oct – 3rd Nov)

Render HMI Graphics

The WinCC HMI design was done in the 3D environment based on the 3D models to achieve a better graphics. The design was carefully based on the designed process and contained all the actual instruments in the laboratory.

Testing Control System

The control system was tested on the actual system in the laboratory. When more shared variables were introduced, the communication delay time was also increased. Hence, it was extremely difficult to control all 3 control loops at the same time. Apart from this, there was no problem in the control system.

Week 15 (4th Nov – 10th Nov)

Laboratory Profibus Network Setup

The Profibus PA network in the ICE laboratory was confirmed on each panel. The Profibus DP network in the ICSE laboratory was also set up according to the design which was able to switch between different network scales.

Project Demonstration

The project outcome was demonstrated to the project supervisor and took about one and half hours. The demonstration illustrated all the significant achievements and outcomes with detailed face to face Q & A.

Preparing Presentation Slides

The presentation slides was designed to include animations to explain the function of the devices, the structure of the network, WinCC HMI screens and the designed networks in the laboratory.

Week 16 (11th Nov – 17th Nov)

Final Presentation

The presentation was at 10:10 a.m. on Thursday and lasted for 12 minutes. Around 50 slides were went though and the feedback was positive.

Organizing Documentations

All the documentation were properly read through for a few times to ensure no important information were missing. All the figures and tables were carefully labelled in the documentation with cross reference.

This is the end of Appendix I.

Appendix II Project Gantt Chart

This section shows a Gantt chart and a timeline which clearly illustrates the project plan and time management.

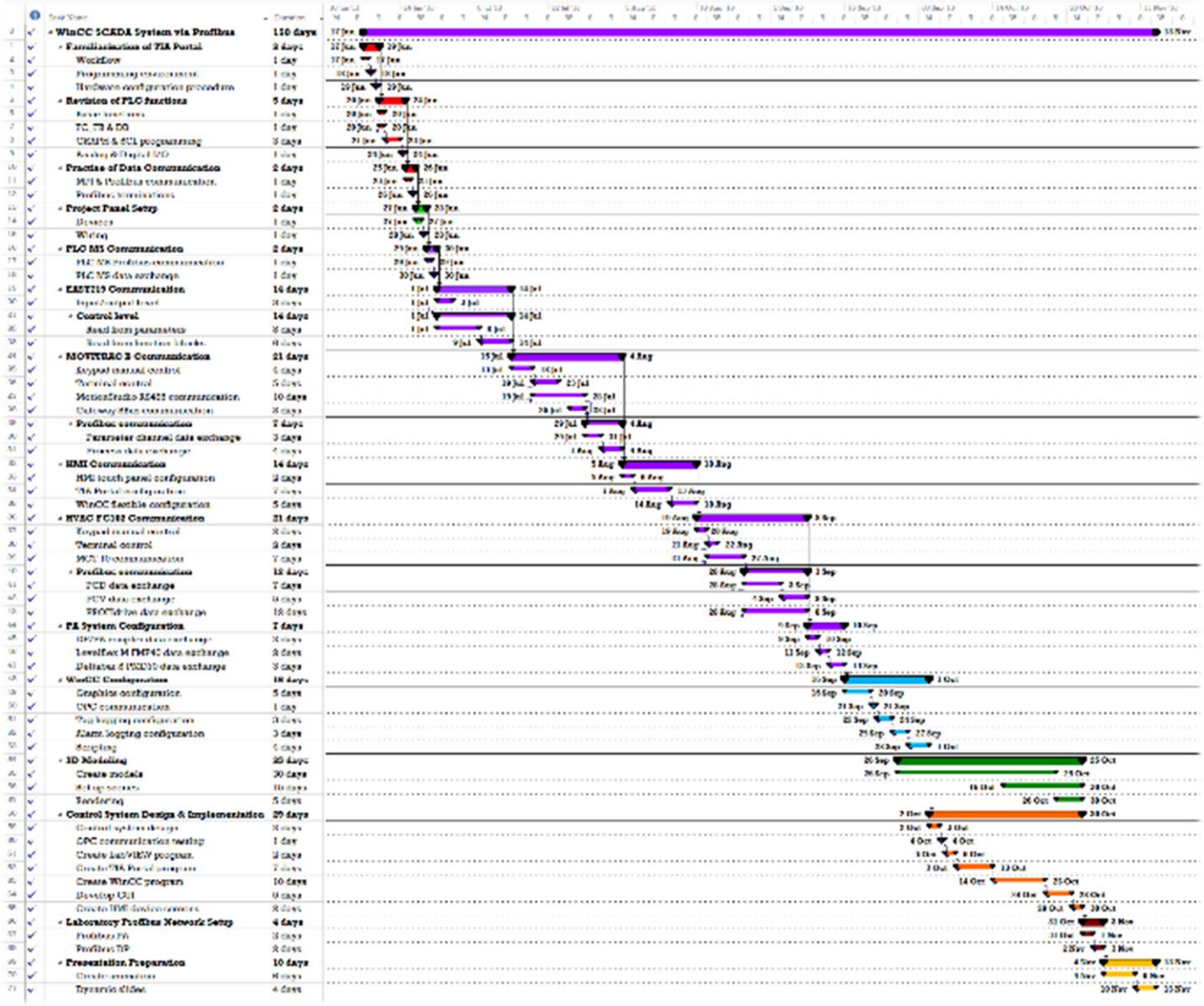


Figure 112: Project Gantt chart

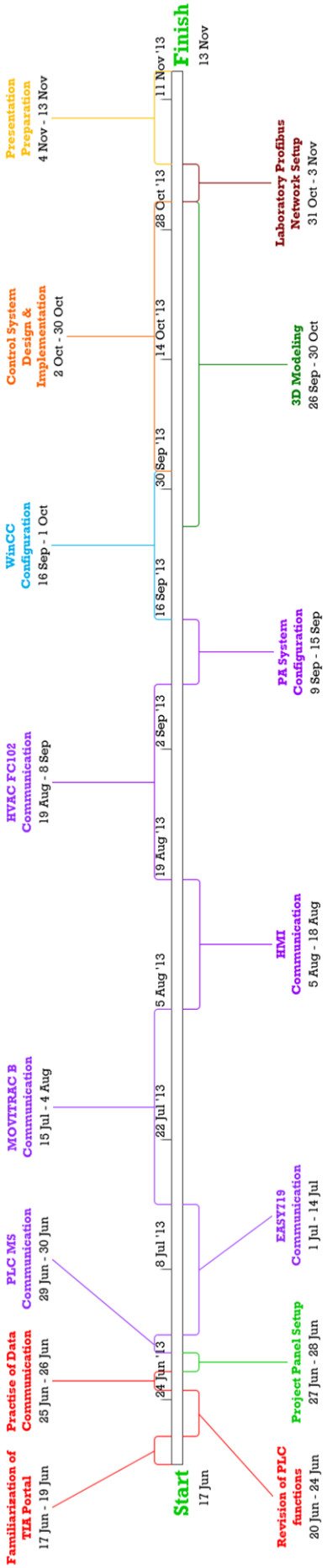


Figure 113: Project timeline

This is the end of Appendix II.

Appendix III Program Code

PLC Code

Primary PLC FB Code

This section contains all the primary PLC FB for the control system in TIA Portal including variable declaration and the actual program. [34] [35] [37] [39]

Moeller [FB1]

Declaration

		Name	Data type	Offset
1		Input		
2		Response	Byte	0.0
3		Level	Real	2.0
4		Temperature	Real	6.0
5		Pressure	Real	10.0
6		Output		
7		ToggleByte	Byte	14.0
8		Command	Byte	15.0
9		Length	Byte	16.0
10		Type	Byte	17.0
11		Index	Byte	18.0
12		Data	Byte	19.0
13		InOut		
14		Static		
15		ToggleInteger	Int	20.0
16		LevelLowBit	Bool	22.0
17		LevelHighBit	Bool	22.1
18		TemperatureLowBit	Bool	22.2
19		TemperatureHighBit	Bool	22.3
20		PressureLowBit	Bool	22.4
21		PressureHighBit	Bool	22.5
22		Feedback	Bool	22.6

Figure 114: Primary PLC Moeller [FB1] declaration in TIA Portal

SCL

```
// Change toggle byte every scan
IF #ToggleInteger >= 5 THEN #ToggleInteger := 0; ELSE #ToggleInteger := #ToggleInteger + 1; END_IF;
IF #ToggleInteger = 0 OR #ToggleInteger = 2 OR #ToggleInteger = 4 THEN #ToggleByte := 16#01;
ELSE #ToggleByte := 16#81; END_IF;

// Constantly send common control bytes
#Command := 16#8C; #Length := 16#01; #Type := 16#86;
// Check process conditions
IF  #Level < 20 THEN #LevelLowBit := 1; ELSE #LevelLowBit := 0; END_IF;
IF  #Level > 80 THEN #LevelHighBit := 1; ELSE #LevelHighBit := 0; END_IF;
IF  #Temperature < 20 THEN #TemperatureLowBit := 1; ELSE #TemperatureLowBit := 0; END_IF;
IF  #Temperature > 80 THEN #TemperatureHighBit := 1; ELSE #TemperatureHighBit := 0; END_IF;
IF  #Pressure < 20 THEN #PressureLowBit := 1; ELSE #PressureLowBit := 0; END_IF;
IF  #Pressure > 80 THEN #PressureHighBit := 1; ELSE #PressureHighBit := 0; END_IF;

// Set outputs depending on the alarm condition
CASE #ToggleInteger OF
  0: #Index := 0; IF #LevelHighBit = 1 THEN #Data := 16#01; ELSE #Data := 16#00; END_IF;
  1: #Index := 1; IF #LevelLowBit = 1 THEN #Data := 16#02; ELSE #Data := 16#00; END_IF;
  2: #Index := 2; IF #TemperatureHighBit = 1 THEN #Data := 16#04; ELSE #Data := 16#00; END_IF;
  3: #Index := 3; IF #TemperatureLowBit = 1 THEN  #Data := 16#08; ELSE #Data := 16#00; END_IF;
  4: #Index := 4; IF #PressureHighBit = 1 THEN  #Data := 16#16; ELSE #Data := 16#00; END_IF;
  5: #Index := 5; IF #PressureLowBit = 1 THEN  #Data := 16#32; ELSE #Data := 16#00; END_IF;
  ELSE: #Index :=6;
  END_CASE;

// Transfer feedback information
IF #Response = 16#C1 THEN #Feedback :=1; ELSE #Feedback := 0; END_IF;
```

Figure 115: Primary PLC Moeller [FB1] SCL code in TIA Portal

SEW [FB2]

Declaration

		Name		Data type	Offset
1		Input			
2		GatewayInfo		DWord	0.0
3		StatusWord		Word	4.0
4		STWBits	AT...	Array [0..15] of Bool	
5		Speed		Word	6.0
6		SpeedPercentage		Word	8.0
7		Response		Byte	10.0
8		ResponseBit	AT...	Array [0..7] of Bool	
9		Output			
10		ToggleByte		Byte	12.0
11		Subindex		Byte	13.0
12		IndexHigh		Byte	14.0
13		IndexLow		Byte	15.0
14		ConrolWord		Word	16.0
15		CTWBits	AT...	Array [0..15] of Bool	
16		MaximumSpeed		Word	18.0
17		SetpointSpeed		Word	20.0
18		InOut			
19		Static			
20		ToggleInteger		Int	22.0
21		FaultStatus		DInt	24.0
22		OperatingHours		DInt	28.0
23		Control		Array [0..15] of Bool	32.0
24		ActualMaximumSpeed		Int	34.0
25		ActualSetpointSpeed		Real	36.0
26		Status		Array [0..15] of Bool	40.0
27		ActualSpeed		Int	42.0
28		ActualSpeedPercentage		Real	44.0
29		Feedback		Bool	48.0

Figure 116: Primary PLC SEW [FB2] declaration in TIA Portal

SCL

```
// Change toggle byte every scan
IF #ToggleInteger >= 1 THEN #ToggleInteger := 0; ELSE #ToggleInteger := #ToggleInteger + 1; END_IF;
IF #ToggleInteger = 0 THEN #ToggleByte := 16#31; ELSE #ToggleByte := 16#71; END_IF;

// Constantly send default index
#Subindex := 16#00;

// Switch between 2 parameters with 1 cycle phase shift
CASE #ToggleInteger OF
  0: #IndexHigh := 16#20; #IndexLow := 16#76; #OperatingHours := DWORD_TO_DINT(#GatewayInfo) / 60;
  1: #IndexHigh := 16#20; #IndexLow := 16#88; #FaultStatus := DWORD_TO_DINT(#GatewayInfo);
END_CASE;

// Transfer feedback information
#Feedback := #ResponseBit[7];

// Transfer CTW
#CTWBits[8] := #Control[0]; #CTWBits[9] := #Control[1]; #CTWBits[10] := #Control[2]; #CTWBits[11] := #Control[3];
#CTWBits[12] := #Control[4]; #CTWBits[13] := #Control[5]; #CTWBits[14] := #Control[6]; #CTWBits[15] := #Control[7];
#CTWBits[0] := #Control[8]; #CTWBits[1] := #Control[9]; #CTWBits[2] := #Control[10]; #CTWBits[3] := #Control[11];
#CTWBits[4] := #Control[12]; #CTWBits[5] := #Control[13]; #CTWBits[6] := #Control[14]; #CTWBits[7] := #Control[15];

// Transfer other process output data
#MaximumSpeed := INT_TO_WORD(#ActualMaximumSpeed * 5);
#SetpointSpeed := DINT_TO_WORD(REAL_TO_DINT(#ActualSetpointSpeed * 5 * 30));

// Transfer STW
#Status[0] := #STWBits[8]; #Status[1] := #STWBits[9]; #Status[2] := #STWBits[10]; #Status[3] := #STWBits[11];
#Status[4] := #STWBits[12]; #Status[5] := #STWBits[13]; #Status[6] := #STWBits[14]; #Status[7] := #STWBits[15];
#Status[8] := #STWBits[0]; #Status[9] := #STWBits[1]; #Status[10] := #STWBits[2]; #Status[11] := #STWBits[3];
#Status[12] := #STWBits[4]; #Status[13] := #STWBits[5]; #Status[14] := #STWBits[6]; #Status[15] := #STWBits[7];

// Transfer other process input data
#ActualSpeed := (WORD_TO_INT(#Speed)) / 5;
#ActualSpeedPercentage := INT_TO_REAL((WORD_TO_INT(#SpeedPercentage))) * 100/16384 ;
```

Figure 117: Primary PLC SEW [FB2] SCL code in TIA Portal

Danfoss [FB3]

Declaration

		Name		Data type	Offset
1		Input			
2		PVARead		DWord	0.0
3		Response		Word	4.0
4		STW		Word	6.0
5		STWBits	AT ...	Array [0..15] of Bool	
6		MAV		Word	8.0
7		Power		Word	10.0
8		Voltage		Word	12.0
9		Frequency		Word	14.0
10		Speed		Word	16.0
11		Current		Word	18.0
12		Temperature		Word	20.0
13		DigitalInputs		Word	22.0
14		DIBits	AT ...	Array [0..15] of Bool	
15		DigitalOutputs		Word	24.0
16		DOBits	AT ...	Array [0..15] of Bool	
17		Output			
18		PCA		Word	26.0
19		RequestBits	AT ...	Array [0..15] of Bool	
20		IND		Byte	28.0
21		PVAWrite		DWord	30.0
22		CTW		Word	34.0
23		CTWBits	AT ...	Array [0..15] of Bool	
24		MRV		Word	36.0
25		LowSpeedLimit		Word	38.0
26		HighSpeedLimit		Word	40.0
27		RampUpTime		Word	42.0
28		RampDownTime		Word	44.0
29		TorqueLimit		Word	46.0
30		DORelayOutput		Word	48.0
31		DORelayOutputBits	AT ...	Array [0..15] of Bool	
32		PulseOutput		Word	50.0
33		AnalogOutput		Word	52.0

		Name	Data type	Offset
32		PulseOutput	Word	50.0
33		AnalogOutput	Word	52.0
34		InOut		
35		Static		
36		ParameterNumber	Int	54.0
37		Request	Int	56.0
38		Index	Int	58.0
39		ActualPVARead	DInt	60.0
40		ActualPVAWrite	DInt	64.0
41		Control	Array [0..15] of Bool	68.0
42		ActualReference	Real	70.0
43		ActualLowSpeedLimit	Int	74.0
44		ActualHighSpeedLimit	Int	76.0
45		ActualRampUpTime	Real	78.0
46		ActualRampDownTime	Real	82.0
47		ActualMotorTorqueLimit	Real	86.0
48		ActualDORelayOutput	Array [0..5] of Bool	90.0
49		ActualPulseOutput	Real	92.0
50		ActualAnalogOutput	Real	96.0
51		Status	Array [0..15] of Bool	100.0
52		ActualSpeedPercentage	Real	102.0
53		ActualPower	Int	106.0
54		ActualMotorVoltage	Real	108.0
55		ActualFrequency	Real	112.0
56		ActualSpeed	Int	116.0
57		ActualMotorCurrent	Real	118.0
58		ActualHeatsinkTemperature	Int	122.0
59		DI	Array [0..5] of Bool	124.0
60		DO	Array [0..1] of Bool	126.0
61		Feedback	Bool	128.0

Figure 118: Primary PLC Danfoss [FB3] declaration in TIA Portal

SCL

```
// Transfer STW
#CTWBits[8] := #Control[0]; #CTWBits[9] := #Control[1]; #CTWBits[10] := #Control[2]; #CTWBits[11] := #Control[3];
#CTWBits[12] := #Control[4]; #CTWBits[13] := #Control[5]; #CTWBits[14] := #Control[6]; #CTWBits[15] := #Control[7];
#CTWBits[0] := #Control[8]; #CTWBits[1] := #Control[9]; #CTWBits[2] := #Control[10]; #CTWBits[3] := #Control[11];
#CTWBits[4] := #Control[12]; #CTWBits[5] := #Control[13]; #CTWBits[6] := #Control[14]; #CTWBits[7] := #Control[15];

// Transfer other PCD write
#MRV := INT_TO_WORD(REAL_TO_INT(#ActualReference * 16384 / 100));
#LowSpeedLimit := INT_TO_WORD(#ActualLowSpeedLimit);
#HighSpeedLimit := INT_TO_WORD(#ActualHighSpeedLimit);
#RampUpTime := INT_TO_WORD(REAL_TO_INT(#ActualRampUpTime * 100));
#RampDownTime := INT_TO_WORD(REAL_TO_INT(#ActualRampDownTime * 100));
#TorqueLimit := INT_TO_WORD(REAL_TO_INT(#ActualMotorTorqueLimit * 100 * 0.1));
#DORelayOutputBits[8] := #ActualDORelayOutput[0]; #DORelayOutputBits[9] := #ActualDORelayOutput[1];
#DORelayOutputBits[12] := #ActualDORelayOutput[4]; #DORelayOutputBits[13] := #ActualDORelayOutput[5];
#PulseOutput := INT_TO_WORD(REAL_TO_INT((#ActualPulseOutput * 163284 / 100) * 0.1));
#AnalogOutput := INT_TO_WORD(REAL_TO_INT((#ActualAnalogOutput * 163284 / 1000) * 0.1));

// Transfer STW
#Status[0] := #STWBits[8]; #Status[1] := #STWBits[9]; #Status[2] := #STWBits[10]; #Status[3] := #STWBits[11];
#Status[4] := #STWBits[12]; #Status[5] := #STWBits[13]; #Status[6] := #STWBits[14]; #Status[7] := #STWBits[15];
#Status[8] := #STWBits[0]; #Status[9] := #STWBits[1]; #Status[10] := #STWBits[2]; #Status[11] := #STWBits[3];
#Status[12] := #STWBits[4]; #Status[13] := #STWBits[5]; #Status[14] := #STWBits[6]; #Status[15] := #STWBits[7];

// Transfer other PCD read
#ActualSpeedPercentage := INT_TO_REAL((WORD_TO_INT(#MAV))) * 100 / 16384;
#ActualPower := (WORD_TO_INT(#Power)) * 10;
#ActualMotorVoltage := INT_TO_REAL(WORD_TO_INT(#Voltage)) *0.1 ;
#ActualFrequency := INT_TO_REAL(WORD_TO_INT(#Frequency)) * 0.1;
#ActualSpeed := WORD_TO_INT(#Speed);
#ActualMotorCurrent := INT_TO_REAL(WORD_TO_INT(#Current)) * 0.01;
#ActualHeatsinkTemperature := (WORD_TO_INT(#Temperature));
#DI[0] := #DIBits[8]; #DI[1] := #DIBits[9]; #DI[2] := #DIBits[10];
#DI[3] := #DIBits[11]; #DI[4] := #DIBits[12]; #DI[5] := #DIBits[13];
#DO[0] := #DOBits[10]; #DO[1] := #DOBits[11];

// Transfer Parameter and index number
#IND := INT_TO_BYTE(#Index);

// Build PCA
#PCA := INT_TO_WORD(#ParameterNumber);
CASE #Request OF
  1: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[4],N := 1); // 0001
  2: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[5],N := 1); // 0010
  3: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[4],N := 2); // 0011
  4: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[6],N := 1); // 0100
  5: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[4],N := 1);
  SET(S_BIT := #RequestBits[6],N := 1); // 0101
  6: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[5],N := 2); // 0110
  7: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[4],N := 3); // 0111
  8: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[7],N := 1); // 1000
  9: RESET(S_BIT := #RequestBits[3],N := 5); SET(S_BIT := #RequestBits[4],N := 1);
  SET(S_BIT := #RequestBits[7],N := 1); // 1001
  ELSE RESET(S_BIT := #RequestBits[3],N := 5);
END_CASE;

// Transfer PVA read
#ActualPVARead := DWORD_TO_DINT(#PVARead);

// Transfer PVA write
#PVAWrite := DINT_TO_DWORD(#ActualPVAWrite);

// Transfer feedback information
IF WORD_TO_INT(SHR(IN := #Response, N := 12)) = 7 THEN #Feedback := 0; ELSE #Feedback := 1;END_IF;
```

Figure 119: Primary PLC Danfoss [FB3] SCL code in TIA Portal

System [FB5]

Declaration

		Name	Data type	Offset
1		Input		
2		Level	Real	0.0
3		Temperature	Real	4.0
4		Pressure	Real	8.0
5		SEWSpeed	Int	12.0
6		DanfossSpeed	Int	14.0
7		PLCStartup	Byte	16.0
8		MoellerStatus	Bool	17.0
9		SEWStatus	Bool	17.1
10		DanfossStatus	Bool	17.2
13		Static		
14		HotLevel	Real	18.0
15		ColdLevel	Real	22.0
16		HotInTemperature	Real	26.0
17		HotOutTemperature	Real	30.0
18		ColdInTemperature	Real	34.0
19		HotFlowRate	Real	38.0
20		ColdFlowRate	Real	42.0
21		HotPump	Bool	46.0
22		ColdPump	Bool	46.1
23		Heater	Bool	46.2
24		PLCStatus	Bool	46.3
25		HMICoordination	Word	48.0
26		HMICoordination... A...	Array [0..15] of Bool	
27		HMIStatus	Bool	50.0
28		HMIHeartBeat	Bool	50.1
29		DPPACouplerStatus	Bool	50.2
30		LevelStatus	Bool	50.3
31		DistanceStatus	Bool	50.4
32		HotLevelStatus	Bool	50.5
33		HotTemperatureStatus	Bool	50.6
34		ColdLevelStatus	Bool	50.7
35		ColdTemperatureStatus	Bool	51.0
36		WarningWord	Word	52.0
37		WarningBits A...	Array [0..15] of Bool	
38		ErrorWord	Word	54.0
39		ErrorBits A...	Array [0..15] of Bool	
40		WarningStatus	DWord	56.0
41		ErrorStatus	DWord	60.0
42		ToggleInteger	Int	64.0
43		Remainder	Int	66.0

Figure 120: Primary PLC System [FB5] declaration in TIA Portal

SCL

```
// Build Warning word
IF #Level < 20 THEN #WarningBits[8] := 1; ELSE #WarningBits[8] := 0; END_IF;
IF #Level > 80 THEN #WarningBits[9] := 1; ELSE #WarningBits[9] := 0; END_IF;
IF #Temperature < 20 THEN #WarningBits[10] := 1; ELSE #WarningBits[10] := 0; END_IF;
IF #Temperature > 80 THEN #WarningBits[11] := 1; ELSE #WarningBits[11] := 0; END_IF;
IF #Pressure < 20 THEN #WarningBits[12] := 1; ELSE #WarningBits[12] := 0; END_IF;
IF #Pressure > 80 THEN #WarningBits[13] := 1; ELSE #WarningBits[13] := 0; END_IF;
IF #HotLevel < 20 THEN #WarningBits[14] := 1; ELSE #WarningBits[14] := 0; END_IF;
IF #HotLevel > 80 THEN #WarningBits[15] := 1; ELSE #WarningBits[15] := 0; END_IF;
IF #ColdLevel < 20 THEN #WarningBits[0] := 1; ELSE #WarningBits[0] := 0; END_IF;
IF #ColdLevel > 80 THEN #WarningBits[1] := 1; ELSE #WarningBits[1] := 0; END_IF;
IF #SEWSpeed < 50 THEN #WarningBits[2] := 1; ELSE #WarningBits[2] := 0; END_IF;
IF #SEWSpeed > 2950 THEN #WarningBits[3] := 1; ELSE #WarningBits[3] := 0; END_IF;
IF #DanfossSpeed < 50 THEN #WarningBits[4] := 1; ELSE #WarningBits[4] := 0; END_IF;
IF #DanfossSpeed > 2950 THEN #WarningBits[5] := 1; ELSE #WarningBits[5] := 0; END_IF;

// Build Error word
IF #PLCStartup = 16#36 THEN #ErrorBits[8] := 0; ELSE #ErrorBits[8] := 1; END_IF;
IF #HMICoordinationBits[8] = 0 THEN #ErrorBits[9] := 0; ELSE #ErrorBits[9] := 1; END_IF;
IF #MoellerStatus = 1 THEN #ErrorBits[10] := 0; ELSE #ErrorBits[10] := 1; END_IF;
IF #SEWStatus = 1 THEN #ErrorBits[11] := 1; ELSE #ErrorBits[11] := 0; END_IF;
IF #DanfossStatus = 1 THEN #ErrorBits[12] := 0; ELSE #ErrorBits[12] := 1; END_IF;
IF #DPPACouplerStatus = 1 THEN #ErrorBits[13] := 0; ELSE #ErrorBits[13] := 1; END_IF;
IF #LevelStatus = 1 THEN #ErrorBits[14] := 0; ELSE  #ErrorBits[14] := 1; END_IF;
IF #DistanceStatus = 1 THEN #ErrorBits[15] := 0; ELSE #ErrorBits[15] := 1; END_IF;
IF #HotLevelStatus = 1 THEN #ErrorBits[0] := 0; ELSE #ErrorBits[0] := 1; END_IF;
IF #HotTemperatureStatus = 1 THEN #ErrorBits[1] := 0; ELSE #ErrorBits[1] := 1; END_IF;
IF #ColdLevelStatus = 1 THEN #ErrorBits[2] := 0; ELSE #ErrorBits[2] := 1; END_IF;
IF #ColdTemperatureStatus = 1 THEN #ErrorBits[3] := 0; ELSE #ErrorBits[3] := 1; END_IF;

// Create dynamic relationship between pressure and the temperature
IF #ToggleInteger > 100 OR #ToggleInteger < 0
THEN #ToggleInteger :=0; ELSE #ToggleInteger := #ToggleInteger + 1; END_IF;

#Remainder := #ToggleInteger MOD 100;
IF #Pressure > #Remainder THEN #Heater := 1; ELSE #Heater := 0; END_IF;

// Transfer PLC and HMI status
IF #PLCStartup = 16#36 THEN #PLCStatus := 1; ELSE #PLCStatus := 0; END_IF;
IF #HMICoordinationBits[8] = 1 THEN #HMIStatus := 1; ELSE #HMIStatus := 0; END_IF;
#HMIHeartBeat := #HMICoordinationBits[10];
```

Figure 121: Primary PLC System [FB5] SCL code in TIA Portal

Toggle [FB100]

Sequence

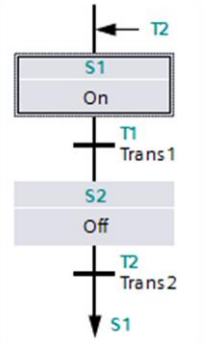


Figure 122: Primary PLC Toggle [FB100] GRAPH code in TIA Portal

Transition

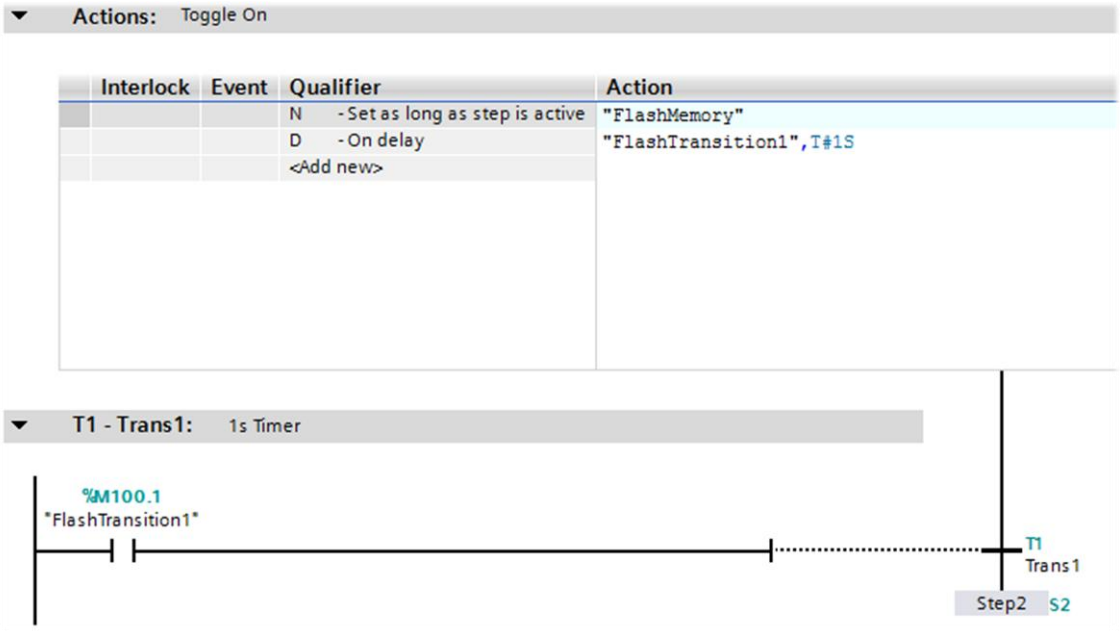


Figure 123: Primary PLC Toggle [FB100] transition 1 code in TIA Portal

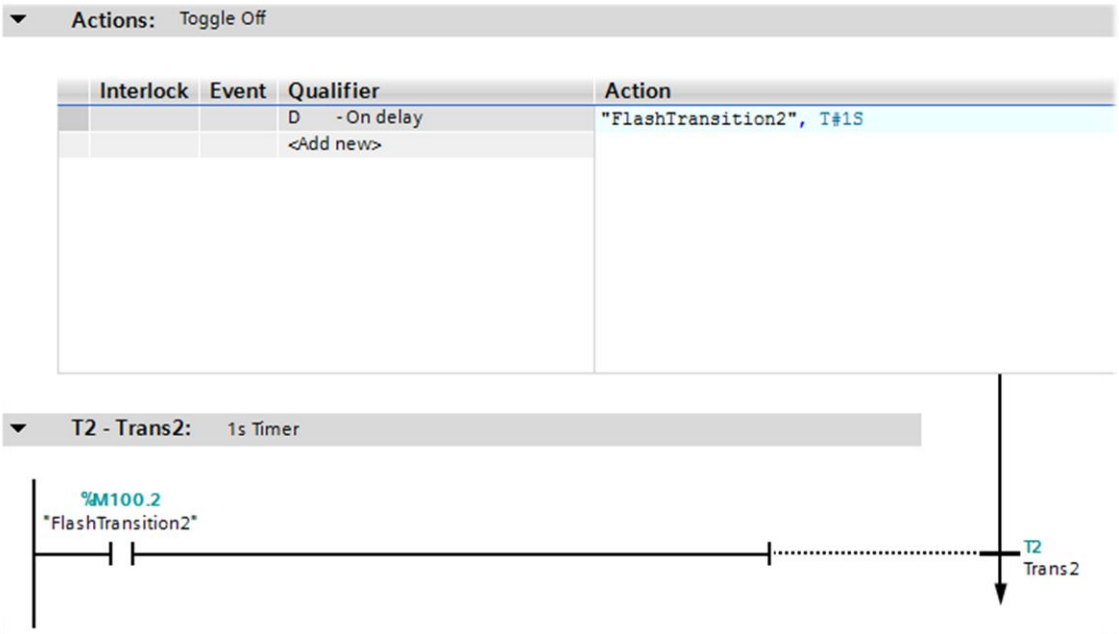


Figure 124: Primary PLC Toggle [FB100] transition 2 code in TIA Portal

Primary PLC OB35 Code

This section contains all the primary PLC OB35 program for the control system in TIA Portal.

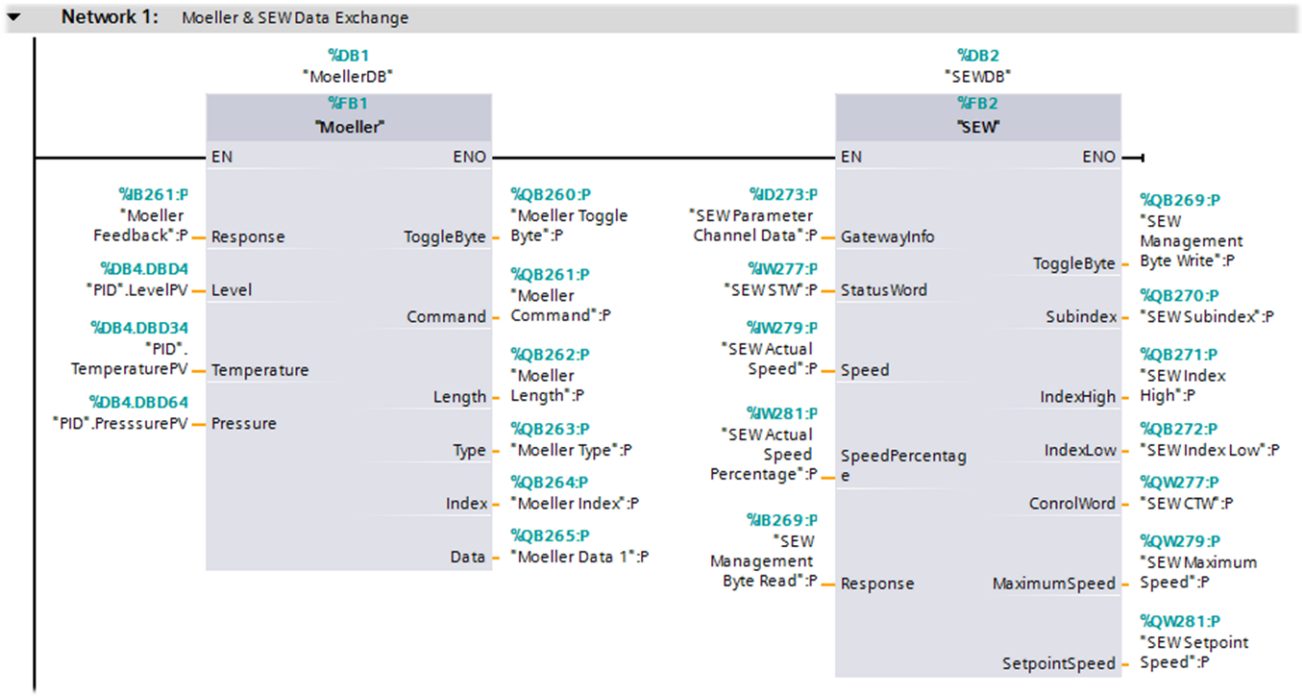


Figure 125: Primary PLC Moeller [FB1] and SEW [FB2] in OB35 in TIA Portal

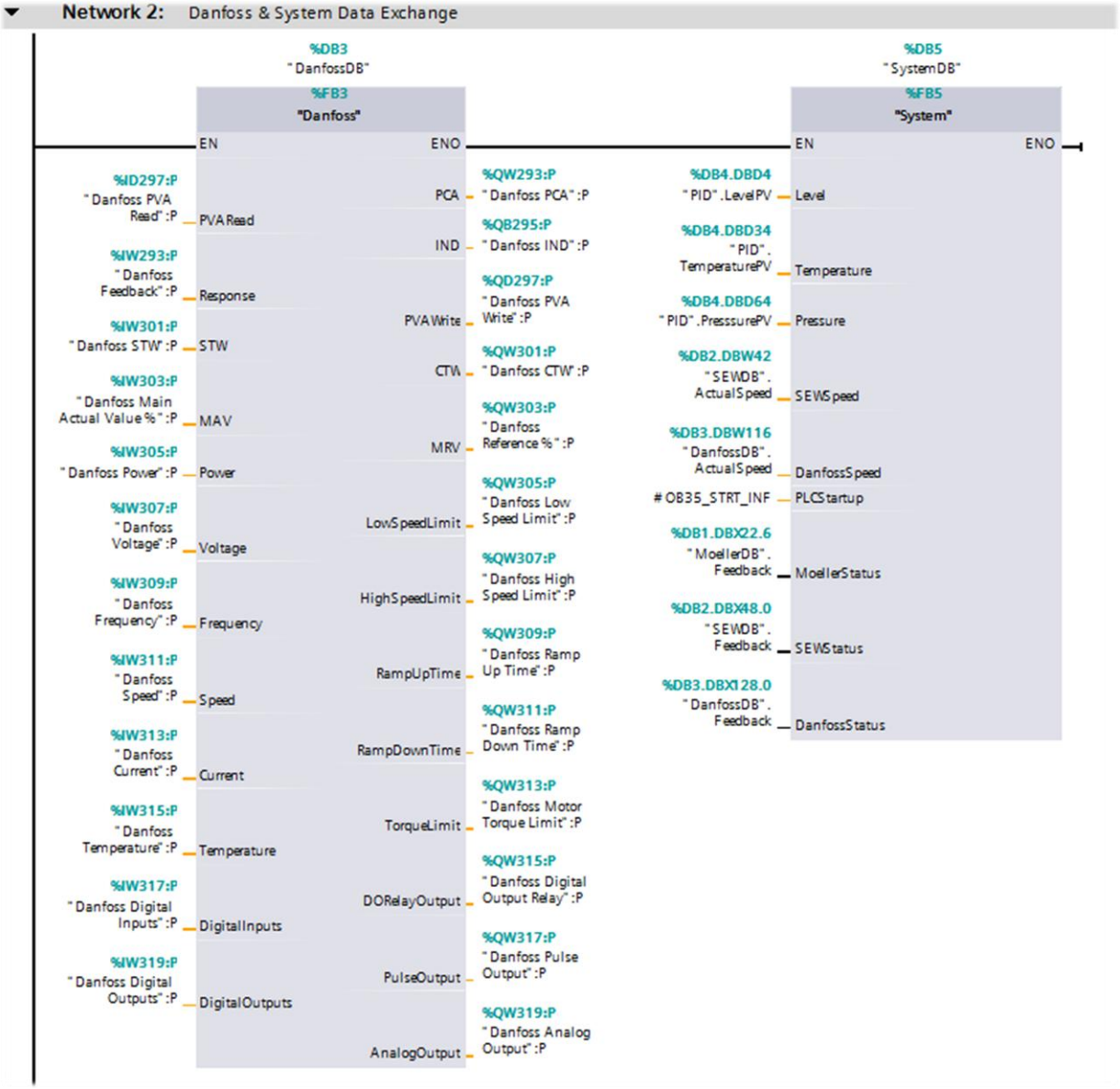


Figure 126: Primary PLC Danfoss [FB3] and System [FB5] in OB35 in TIA Portal

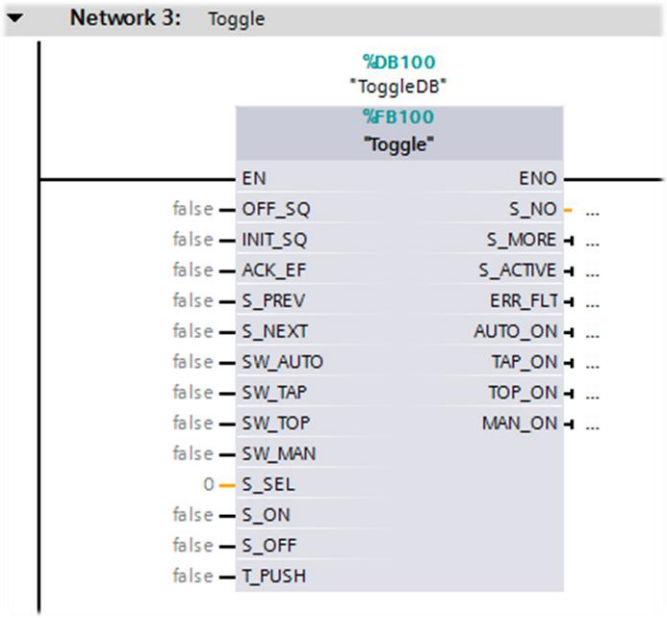


Figure 127: Primary PLC Toggle [FB100] in OB35 in TIA Portal

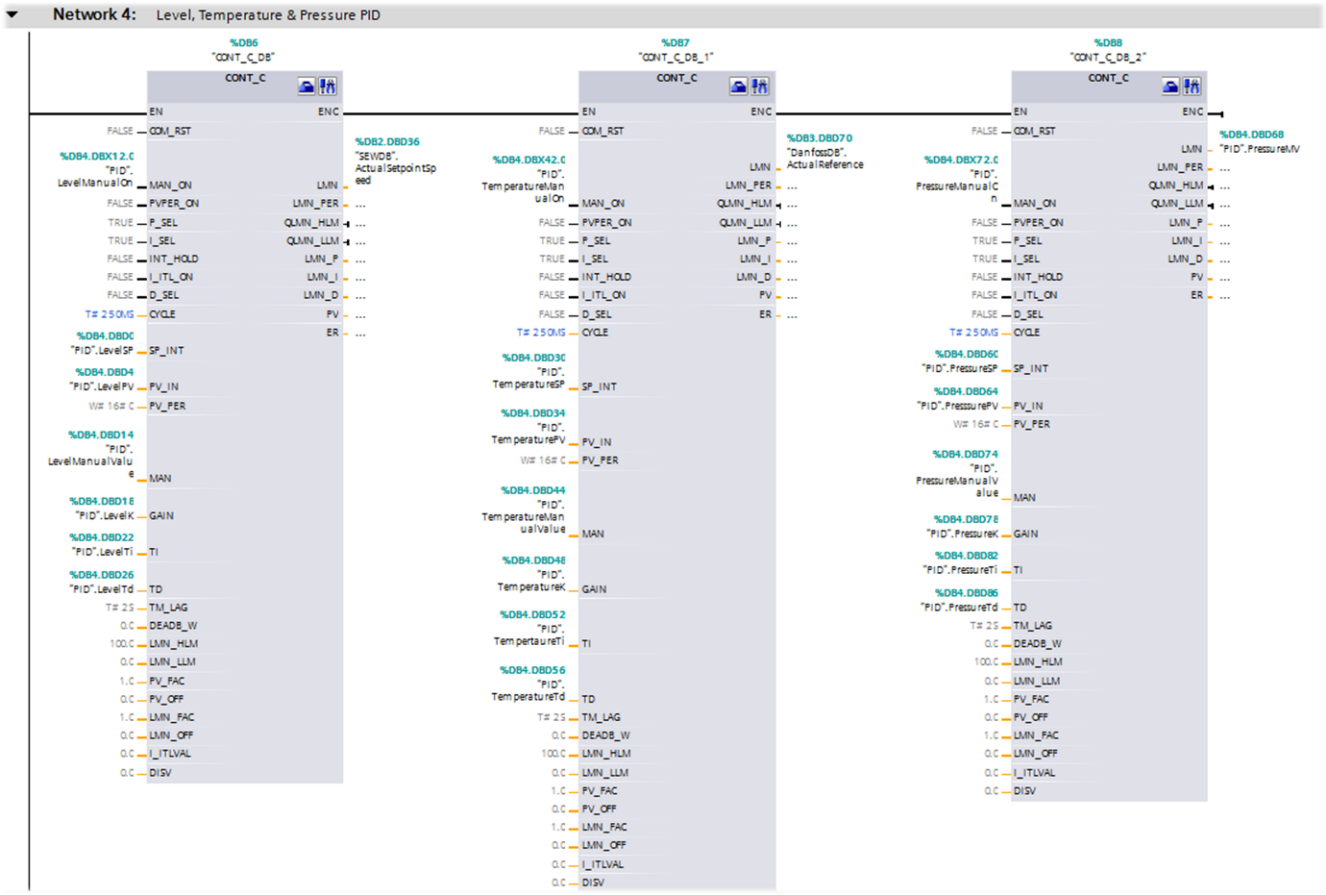


Figure 128: Primary PLC PID controller blocks in OB35 in TIA Portal

Secondary PLC FB Code

This section contains all the secondary PLC FB for the control system in TIA Portal including variable declaration and the actual program. [34] [35] [36] [37] [39]

DPPACoupler [FB6]

Declaration

		Name		Data type	Offset
1		Input			
2		CurrentRawData		Word	0.0
3		CurrentRawBits	AT...	Array [0..15] of Bool	
4		CurrentStatus		Byte	2.0
5		Output			
6		InOut			
7		Static			
8		ActualCurrent		Real	4.0
9		Feedback		Bool	8.0
10		Temp			
11		CurrentVZ		Int	0.0
12		CurrentExponent		Int	2.0
13		CurrentFraction		Byte	4.0
14		CurrentFractionBits	AT...	Array [0..7] of Bool	

Figure 129: Secondary PLC DPPACoupler [FB6] declaration in TIA Portal

SCL

```
1  // Obtain Current exponent value
2  #CurrentExponent := WORD_TO_INT(SHR(IN := SHL(IN := #CurrentRawData, N := 1), N := 8));
3
4  // Obtian Current fraction byte
5  #CurrentFraction := WORD_TO_BYTE (SHR(IN := SHL(IN := #CurrentRawData, N := 9), N := 9));
6
7  // Obtain Current VZ value
8  #CurrentVZ := BOOL_TO_INT(#CurrentRawBits[7]);
9
10 // Calculate actual value
11 #ActualCurrent := ((-1)**#CurrentVZ) * (2**(#CurrentExponent - 127)) * (1 + (
12 BOOL_TO_INT(#CurrentFractionBits[6])*0.5 +
13 BOOL_TO_INT(#CurrentFractionBits[5])*0.25 +
14 BOOL_TO_INT(#CurrentFractionBits[4])*0.125 +
15 BOOL_TO_INT(#CurrentFractionBits[3])*0.0625 +
16 BOOL_TO_INT(#CurrentFractionBits[2])*0.03125 +
17 BOOL_TO_INT(#CurrentFractionBits[1])*0.015625 +
18 BOOL_TO_INT(#CurrentFractionBits[0])*0.00781235));
19
20 IF #CurrentStatus = 16#80 THEN #Feedback := 1; ELSE #Feedback := 0; END_IF;
```

Figure 130: Secondary PLC DPPACoupler [FB6] SCL code in TIA Portal

LevelTransmitter [FB7]

Declaration

		Name		Data type	Offset
1		Input			
2		LevelRawData		Word	0.0
3		LevelRawBits	AT ...	Array [0..15] of Bool	
4		LevelStatus		Byte	2.0
5		DistanceRawData		Word	4.0
6		DistanceRawBits	AT ...	Array [0..15] of Bool	
7		DistanceStatus		Byte	6.0
8		Output			
9		InOut			
10		Static			
11		ActualLevel		Real	8.0
12		LevelFeedback		Bool	12.0
13		ActualDistance		Real	14.0
14		DistanceFeedback		Bool	18.0
15		Temp			
16		LevelVZ		Int	0.0
17		LevelExponent		Int	2.0
18		LevelFraction		Byte	4.0
19		LevelFractionBits	AT ...	Array [0..7] of Bool	
20		DistanceVZ		Int	6.0
21		DistanceExponent		Int	8.0
22		DistanceFraction		Byte	10.0
23		DistanceFractionBits	AT ...	Array [0..7] of Bool	

Figure 131: Secondary PLC LevelTransmitter [FB7] declaration in TIA Portal

SCL

```
// Obtain Level exponent value
#LevelExponent := WORD_TO_INT(SHR(IN := SHL(IN := #LevelRawData, N := 1), N := 8));

// Obtian Level fraction byte
#LevelFraction := WORD_TO_BYTE (SHR(IN := SHL(IN := #LevelRawData, N := 9), N := 9));

// Obtain Level VZ value
#LevelVZ := BOOL_TO_INT(#LevelRawBits[7]);

// Calculate actual value
#ActualLevel := ((-1)**#LevelVZ) * (2**(#LevelExponent - 127)) * (1 + (
BOOL_TO_INT(#LevelFractionBits[6])*0.5 +
BOOL_TO_INT(#LevelFractionBits[5])*0.25 +
BOOL_TO_INT(#LevelFractionBits[4])*0.125 +
BOOL_TO_INT(#LevelFractionBits[3])*0.0625 +
BOOL_TO_INT(#LevelFractionBits[2])*0.03125 +
BOOL_TO_INT(#LevelFractionBits[1])*0.015625 +
BOOL_TO_INT(#LevelFractionBits[0])*0.00781235));

// Transfer feedback information
IF #LevelStatus = 16#80 THEN #LevelFeedback := 1; ELSE #LevelFeedback := 0; END_IF;


// Obtain Distance value
#DistanceExponent := WORD_TO_INT(SHR(IN := SHL(IN := #DistanceRawData, N := 1), N := 8));

// Obtian Distance fraction byte
#DistanceFraction := WORD_TO_BYTE (SHR(IN := SHL(IN := #DistanceRawData, N := 9), N := 9));

// Obtain Distance VZ value
#DistanceVZ := BOOL_TO_INT(#DistanceRawBits[7]);

// Calculate actual value
#ActualDistance := ((-1)**#DistanceVZ) * (2**(#DistanceExponent - 127)) * (1 + (
BOOL_TO_INT(#DistanceFractionBits[6])*0.5 +
BOOL_TO_INT(#DistanceFractionBits[5])*0.25 +
BOOL_TO_INT(#DistanceFractionBits[4])*0.125 +
BOOL_TO_INT(#DistanceFractionBits[3])*0.0625 +
BOOL_TO_INT(#DistanceFractionBits[2])*0.03125 +
BOOL_TO_INT(#DistanceFractionBits[1])*0.015625 +
BOOL_TO_INT(#DistanceFractionBits[0])*0.00781235));

// Transfer feedback information
IF #DistanceStatus = 16#80 THEN #DistanceFeedback := 1; ELSE #DistanceFeedback := 0; END_IF;
```

Figure 132: Secondary PLC LevelTransmitter [FB7] SCL code in TIA Portal

PressureTransmitterHot [FB8]

Declaration

		Name		Data type	Offset
1		Input			
2		PressureHotRawData		Word	0.0
3		PressureHotRawBits	AT "P...	Array [0..15] of Bool	
4		PressureHotStatus		Byte	2.0
5		TemperatureHotRawData		Word	4.0
6		TemperatureHotRawBits	AT "T...	Array [0..15] of Bool	
7		TemperatureHotStatus		Byte	6.0
8		Output			
9		InOut			
10		Static			
11		ScaledHotLevel		Real	8.0
12		LevelFeedback		Bool	12.0
13		ActualHotTemperature		Real	14.0
14		TemperatureFeedback		Bool	18.0
15		Temp			
16		PressureHotVZ		Int	0.0
17		PressureHotExponent		Int	2.0
18		PressureHotFraction		Byte	4.0
19		PressureHotFractionBits	AT "P...	Array [0..7] of Bool	
20		ActualHotLevel		Real	6.0
21		TemperatureHotVZ		Int	10.0
22		TemperatureHotExponent		Int	12.0
23		TemperatureHotFraction		Byte	14.0
24		TemperatureHotFractionBits	AT "T...	Array [0..7] of Bool	

Figure 133: Secondary PLC PressureTransmitterHot [FB8] declaration in TIA Portal

SCL

```
// Obtain PressureHot exponent value
#PressureHotExponent := WORD_TO_INT(SHR(IN := SHL(IN := #PressureHotRawData, N := 1), N := 8));

// Obtian PressureHot fraction byte
#PressureHotFraction := WORD_TO_BYTE (SHR(IN := SHL(IN := #PressureHotRawData, N := 9), N := 9));

// Obtain PressureHot VZ value
#PressureHotVZ := BOOL_TO_INT(#PressureHotRawBits[7]);

// Calcualte actual value
#ActualHotLevel := ((-1)**#PressureHotVZ) * (2**(#PressureHotExponent - 127)) * (1 + (
BOOL_TO_INT(#PressureHotFractionBits[6])*0.5 +
BOOL_TO_INT(#PressureHotFractionBits[5])*0.25 +
BOOL_TO_INT(#PressureHotFractionBits[4])*0.125 +
BOOL_TO_INT(#PressureHotFractionBits[3])*0.0625 +
BOOL_TO_INT(#PressureHotFractionBits[2])*0.03125 +
BOOL_TO_INT(#PressureHotFractionBits[1])*0.015625 +
BOOL_TO_INT(#PressureHotFractionBits[0])*0.00781235));

// Scale the calculated value
#ScaledHotLevel := #ActualHotLevel * -4 ;

// Transfer feedback infomration
IF #PressureHotStatus = 16#80 THEN #LevelFeedback := 1; ELSE #LevelFeedback := 0; END_IF;


// Obtain TemperatureHot exponent value
#TemperatureHotExponent := WORD_TO_INT(SHR(IN := SHL(IN := #TemperatureHotRawData, N := 1), N := 8));

// Obtian TemperatureHot fraction byte
#TemperatureHotFraction := WORD_TO_BYTE (SHR(IN := SHL(IN := #TemperatureHotRawData, N := 9), N := 9));

// Obtain TemperatureHot VZ value
#TemperatureHotVZ := BOOL_TO_INT(#TemperatureHotRawBits[7]);

// Calculate actual value
#ActualHotTemperature := ((-1)**#TemperatureHotVZ) * (2**(#TemperatureHotExponent - 127)) * (1 + (
BOOL_TO_INT(#TemperatureHotFractionBits[6])*0.5 +
BOOL_TO_INT(#TemperatureHotFractionBits[5])*0.25 +
BOOL_TO_INT(#TemperatureHotFractionBits[4])*0.125 +
BOOL_TO_INT(#TemperatureHotFractionBits[3])*0.0625 +
BOOL_TO_INT(#TemperatureHotFractionBits[2])*0.03125 +
BOOL_TO_INT(#TemperatureHotFractionBits[1])*0.015625 +
BOOL_TO_INT(#TemperatureHotFractionBits[0])*0.00781235));

// Transfer feedback information
IF #TemperatureHotStatus = 16#80 THEN #TemperatureFeedback := 1; ELSE #TemperatureFeedback := 0; END_IF;
```

Figure 134: Secondary PLC PressureTransmitterHot [FB8] SCL code in TIA Portal

PressureTransmitterCold [FB9]

Declaration

		Name		Data type	Offset
1		Input			
2		PressureColdRawData		Word	0.0
3		PressureColdRawBits	AT*...	Array [0..15] of Bool	
4		PressureColdStatus		Byte	2.0
5		TemperatureColdRawData		Word	4.0
6		TemperatureColdRawBits	AT*...	Array [0..15] of Bool	
7		TemperatureColdStatus		Byte	6.0
8		Output			
9		InOut			
10		Static			
11		ScaledColdLevel		Real	8.0
12		LevelFeedback		Bool	12.0
13		ActualColdTemperature		Real	14.0
14		TemperatureFeedback		Bool	18.0
15		Temp			
16		PressureColdVZ		Int	0.0
17		PressureColdExponent		Int	2.0
18		PressureColdFraction		Byte	4.0
19		PressureColdFractionBits	AT*...	Array [0..7] of Bool	
20		ActualColdLevel		Real	6.0
21		TemperatureColdVZ		Int	10.0
22		TemperatureColdExponent		Int	12.0
23		TemperatureColdFraction		Byte	14.0
24		TemperatureColdFractionBits	AT*...	Array [0..7] of Bool	

Figure 135: Secondary PLC PressureTransmitterCold [FB9] declaration in TIA Portal

SCL

```
// Obtain PressureCold exponent value
#PressureColdExponent := WORD_TO_INT(SHR(IN := SHL(IN := #PressureColdRawData, N := 1), N := 8));

// Obtian PressureCold fraction byte
#PressureColdFraction := WORD_TO_BYTE (SHR(IN := SHL(IN := #PressureColdRawData, N := 9), N := 9));

// Obtain PressureCold VZ value
#PressureColdVZ := BOOL_TO_INT(#PressureColdRawBits[7]);

// Calcualte actual level
#ActualColdLevel := ((-1)**#PressureColdVZ) * (2**(#PressureColdExponent - 127)) * (1 + (
BOOL_TO_INT(#PressureColdFractionBits[6])*0.5 +
BOOL_TO_INT(#PressureColdFractionBits[5])*0.25 +
BOOL_TO_INT(#PressureColdFractionBits[4])*0.125 +
BOOL_TO_INT(#PressureColdFractionBits[3])*0.0625 +
BOOL_TO_INT(#PressureColdFractionBits[2])*0.03125 +
BOOL_TO_INT(#PressureColdFractionBits[1])*0.015625 +
BOOL_TO_INT(#PressureColdFractionBits[0])*0.00781235));

// Scale the calculated value
#ScaledColdLevel := #ActualColdLevel * -4;

// Transfer feedback information
IF #PressureColdStatus = 16#80 THEN #LevelFeedback := 1; ELSE #LevelFeedback := 0; END_IF;


// Obtain TemperatureCold exponent value
#TemperatureColdExponent := WORD_TO_INT(SHR(IN := SHL(IN := #TemperatureColdRawData, N := 1), N := 8));

// Obtian TemperatureCold fraction byte
#TemperatureColdFraction := WORD_TO_BYTE (SHR(IN := SHL(IN := #TemperatureColdRawData, N := 9), N := 9));

// Obtain TemperatureCold VZ value
#TemperatureColdVZ := BOOL_TO_INT(#TemperatureColdRawBits[7]);

// Calculate actual temperature
#ActualColdTemperature := ((-1)**#TemperatureColdVZ) * (2**(#TemperatureColdExponent - 127)) * (1 + (
BOOL_TO_INT(#TemperatureColdFractionBits[6])*0.5 +
BOOL_TO_INT(#TemperatureColdFractionBits[5])*0.25 +
BOOL_TO_INT(#TemperatureColdFractionBits[4])*0.125 +
BOOL_TO_INT(#TemperatureColdFractionBits[3])*0.0625 +
BOOL_TO_INT(#TemperatureColdFractionBits[2])*0.03125 +
BOOL_TO_INT(#TemperatureColdFractionBits[1])*0.015625 +
BOOL_TO_INT(#TemperatureColdFractionBits[0])*0.00781235));

// Transfer feedback information
IF #TemperatureColdStatus = 16#80 THEN #TemperatureFeedback := 1; ELSE #TemperatureFeedback := 0; END_IF;
```

Figure 136: Secondary PLC PressureTransmitterCold [FB9] SCL code in TIA Portal

Secondary PLC OB35 Code

This section contains all the secondary PLC OB35 program for the control system in TIA Portal.

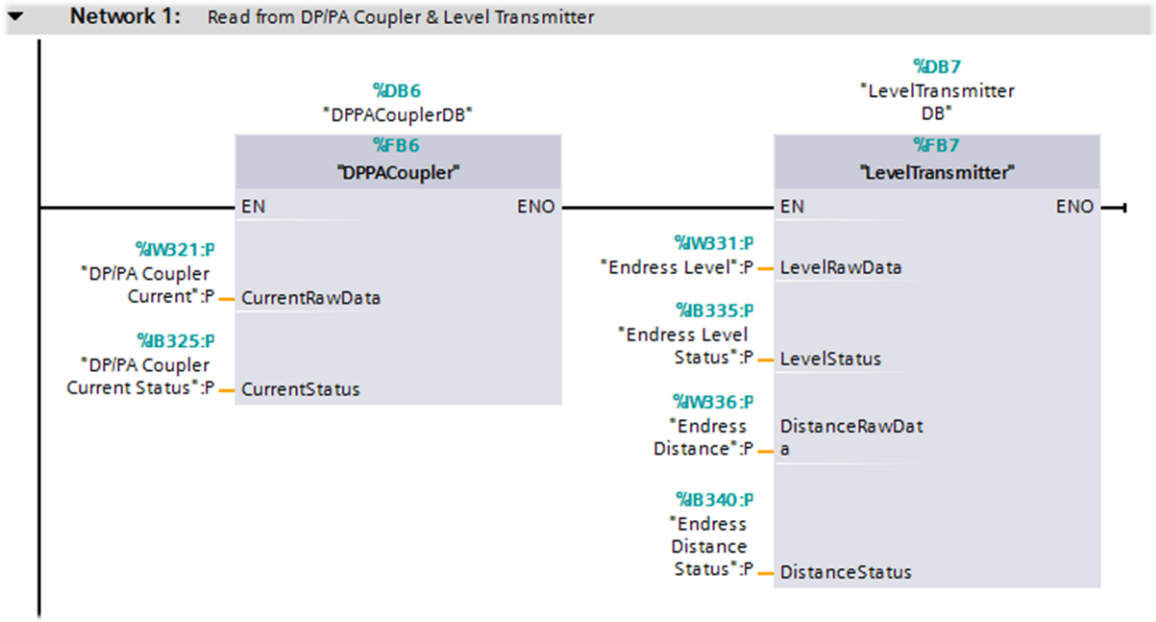


Figure 137: Secondary PLC DPPACoupler [FB6] and LevelTransmitter [FB7] in OB35 in TIA Portal

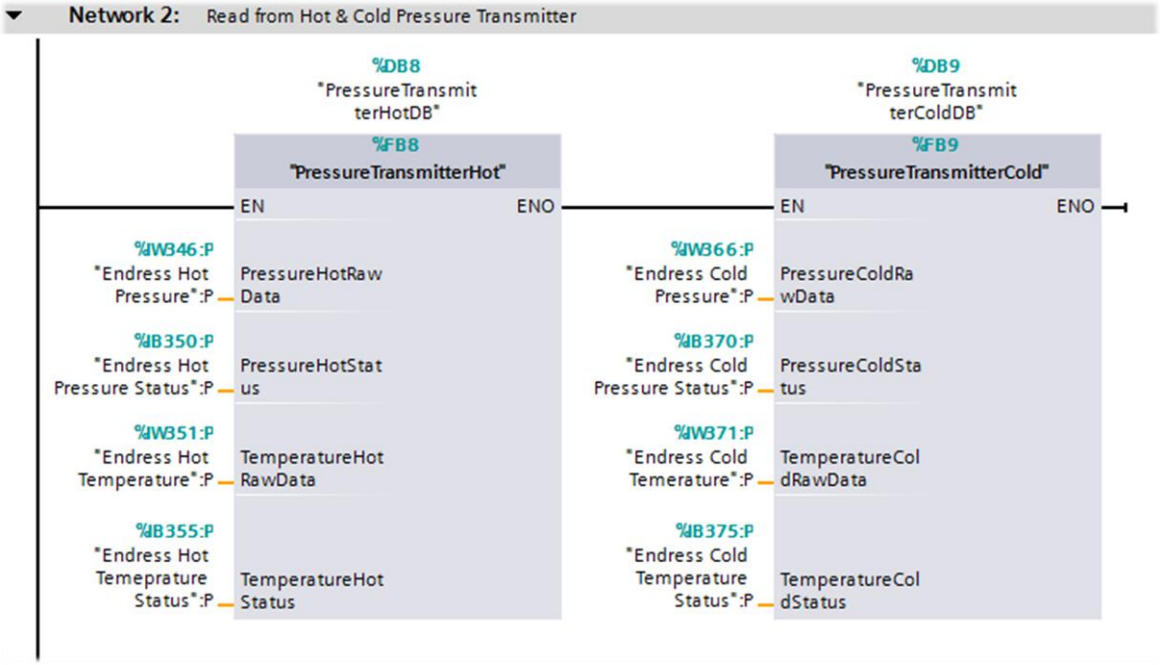


Figure 138: Secondary PLC PressureTransmitterHot [FB8] and PressureTransmitterCold [FB9] in OB35 in TIA Portal

WinCC Code

This section contains all the typical code for the control system in WinCC including ANSI-C, VBScript, VBA and SQL code. [112]

ANSI-C

Toggle State

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{

if (GetTagBit("PIDTemperatureManualOn") ==1 )
{

SetTagBit("PIDTemperatureManualOn",0);
}

else {
   (SetTagBit("PIDTemperatureManualOn",1));
}

printf ("Change PID temperature auto/manual state \r\n");

}
```

Figure 139: ANSI-C script to toggle state of a binary tag in WinCC

Transfer OPC Value to PLC

```
#include "apdefap.h"

int gscAction( void )
{

#pragma option(mbcs)

SetTagFloat("HotOutTemperature",GetTagFloat("OPCHotOutTemperature"));

printf ("Successfully send hot output temperature from LabVIEW to PLC\r\n");

return 0;

}
```

Figure 140: ANSI-C script to transfer values between different tags in WinCC

VBS

Animate Colour of I/O Field Value

```
Function ForeColor_Trigger(Byval Item)

    Dim objTag

    Set objTag = HMIRuntime.Tags("OPCPressureValve")
    objTag.Read

    If objTag.Value < 20 Then
    ForeColor_Trigger = RGB(0,0,255)

    Elseif objtag.Value > 20 And objtag.Value < 80 Then
    ForeColor_Trigger = RGB(0,255,0)

    Else
    ForeColor_Trigger = RGB(255,0,0)

    End If

    HMIRuntime.Trace "Pressure valve value range changed" & vbCrLf

End Function
```

Figure 141: VBScript to dynamically change colour of an I/O field display in WinCC

Animate Tank Fill Level

```
Function FillingIndex_Trigger(ByVal Item)

    Dim objTag

    Set objTag = HMIRuntime.Tags("EndressHotLevel")
    objTag.Read

    FillingIndex_Trigger = objTag.Value

    HMIRuntime.Trace "Hot level changed" & vbCrLf

End Function
```

Figure 142: VBScript to display the object length according to a tag value in WinCC

Animate Visibility

```
Function Visible_Trigger(ByVal Item)

Dim objTag

Set objTag = HMIRuntime.Tags("FlashMemory")
objTag.Read

If objTag.Value = 1 Then
Visible_Trigger = 0

Else
Visible_Trigger = 1

End If

HMIRuntime.Trace "Temperature alarm is activated" & vbCrLf

End Function
```

Figure 143: VBScript to animate the visibility of an object according to a binary tag value in WinCC

Animate Transparency

```
Function Visible_Trigger(ByVal Item)

    Dim objTag

    Set objTag = HMIRuntime.Tags("FlashMemory")
    objTag.Read

    If objTag.Value = 1 Then
    Visible_Trigger = 0

    Else
    Visible_Trigger = 1

    End If

    HMIRuntime.Trace "Level alarm is activated" & vbCrLf

End Function
```

Figure 144: VBScript to animate the transparency of an object according to a binary tag value in WinCC

Load Screen to Picture Window

```
Sub OnLButtonUp(ByVal Item, ByVal Flags, ByVal x, ByVal y)

    Dim objScrWindow

    Set objScrWindow = ScreenItems("Picture Loader")

    objScrWindow.Visible = 0

    HMIRuntime.Trace "Process screen is loaded" & vbCrLf

End Sub
```

Figure 145: VBScript to load screen into the picture window in WinCC

Stop Runtime

```
Sub OnLButtonUp(Byval Item, Byval Flags, Byval x, Byval y)

    HMIRuntime.Stop

    HMIRuntime.Trace "Runtime is stopped" & vbCrLf

End Sub
```

Figure 146: VBScript to stop Runtime in WinCC

Excel VBA Program to Access WinCC OPC Tags

```
Option Explicit
Option Base 1
Const ServerName = "OPCServer.WinCC"
Dim WithEvents MyOPCServer As OPCServer
Dim WithEvents MyOPCGroup As OPCGroup
Dim MyOPCGroupColl As OPCGroups
Dim MyOPCItemColl As OPCItems
Dim MyOPCItems As OPCItems
Dim MyOPCItem As OPCItem
Dim ClientHandles(2) As Long
Dim ServerHandles() As Long
Dim Values(2) As Variant
Dim Errors() As Long
Dim ItemIDs(2) As String
Dim GroupName As String
Dim NodeName As String
Dim Tag(2) As Variant
Dim i As Integer
```

Figure 147: Declare variables

```
 ' Sub to connect to OPC server
Sub StartClient()

 ' Create ClientHandles
For i = 1 To 2
 ClientHandles(i) = i
Next i

 ' Create a group
 GroupName = "MyGroup"

 ' Define the computer name
 NodeName = Range("A56").Value

 ' Define tag names
 ItemIDs(1) = Range("A2").Value: ItemIDs(2) = Range("A3").Value

 ' Create a OPC instance
 Set MyOPCServer = New OPCServer
 MyOPCServer.Connect ServerName, NodeName
 Set MyOPCGroupColl = MyOPCServer.OPCGroups

 ' Set the default active state for adding groups
 MyOPCGroupColl.DefaultGroupIsActive = True

 ' Add group to the Collection
 Set MyOPCGroup = MyOPCGroupColl.Add(GroupName)
 Set MyOPCItemColl = MyOPCGroup.OPCItems

 ' Add items, return ServerHandles
 MyOPCItemColl.AddItems 2, ItemIDs(), ClientHandles(), ServerHandles(), Errors

 ' A group that is subscribed receives asynchronous notifications
 MyOPCGroup.IsSubscribed = True
 Exit Sub

ErrorHandler:
 MsgBox "Error: " & Err.Description, vbCritical, "ERROR"

End Sub
```

Figure 148: Sub to connect to WinCC OPC server

```
' Sub to disconnect from OPC server
Sub StopClient()

' Release group
MyOPCGroupColl.RemoveAll

' Clear objects
MyOPCServer.Disconnect
Set MyOPCItemColl = Nothing
Set MyOPCGroup = Nothing
Set MyOPCGroupColl = Nothing
Set MyOPCServer = Nothing

End Sub
```

Figure 149: Sub to disconnect from WinCC OPC server

```
' Read tag values and write to cells
Private Sub MyOPCGroup_DataChange(ByVal TransactionID As Long, ByVal NumItems As Long, _
ClientHandles() As Long, TagValues() As Variant, Qualities() As Long, TimeStamps() As Date)

For i = 1 To NumItems
 Tag(ClientHandles(i)) = TagValues(i)
Next i

 Range("I56").Value = CStr(TimeStamps(1))
 Range("B2").Value = CStr(Tag(1)): Range("B3").Value = CStr(Tag(2))

End Sub
```

Figure 150: Sub to read WinCC OPC tags

```
' Write to tags upon cell change
Private Sub worksheet_change(ByVal Target As Range)

  If Not Intersect(Target, Range("D2")) Is Nothing Then Values(1) = Range("D2").Value
  MyOPCGroup.SyncWrite 1, ServerHandles, Values, Errors

  If Not Intersect(Target, Range("D3")) Is Nothing Then Values(2) = Range("D3").Value
  MyOPCGroup.SyncWrite 2, ServerHandles, Values, Errors

End Sub
```

Figure 151: Sub to write to WinCC OPC tags

SQL program to Select Data after Step Change

```
SELECT [Data Table].[Level Time], [Data Table].[Level SP], [Data Table].[Level MV], [Data Table].[Level PV]
FROM [Data Table]
WHERE ((([Data Table].[Level SP])>50));
```

Figure 152: SQL code to make a query to obtain the data after the step change was made in Access

This is the end of Appendix III.

Appendix IV Gallery

This section contains all the 3D model presets designed for the project for illustration purposes.

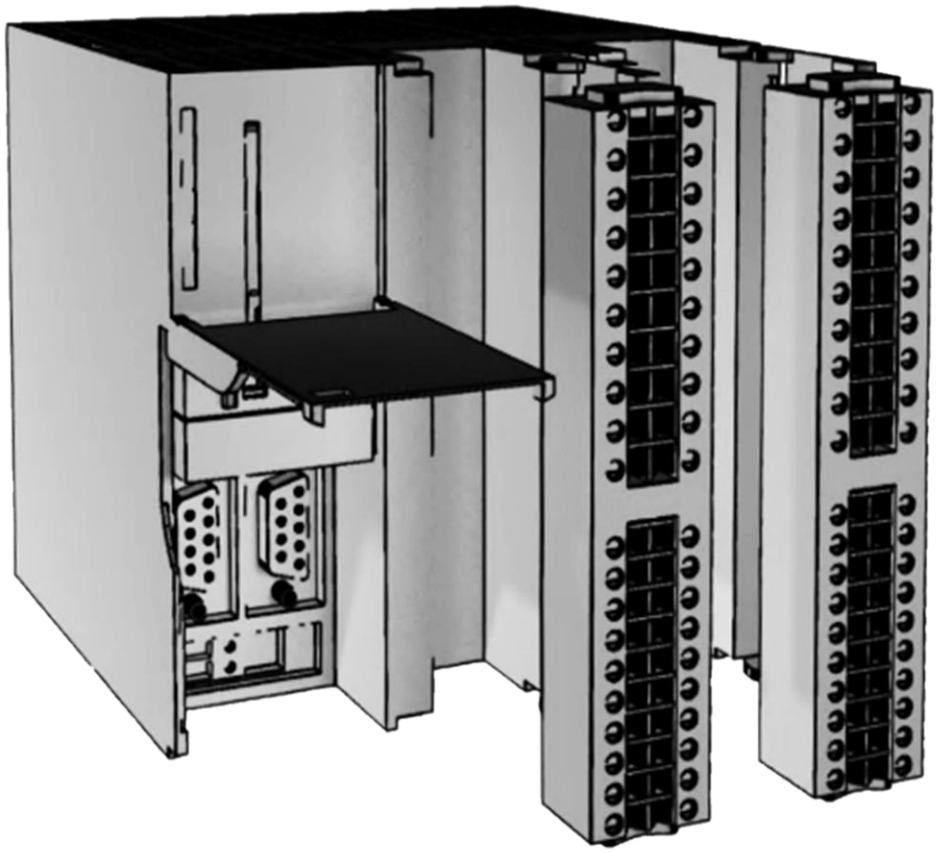


Figure 153: S7-300 PLC preset

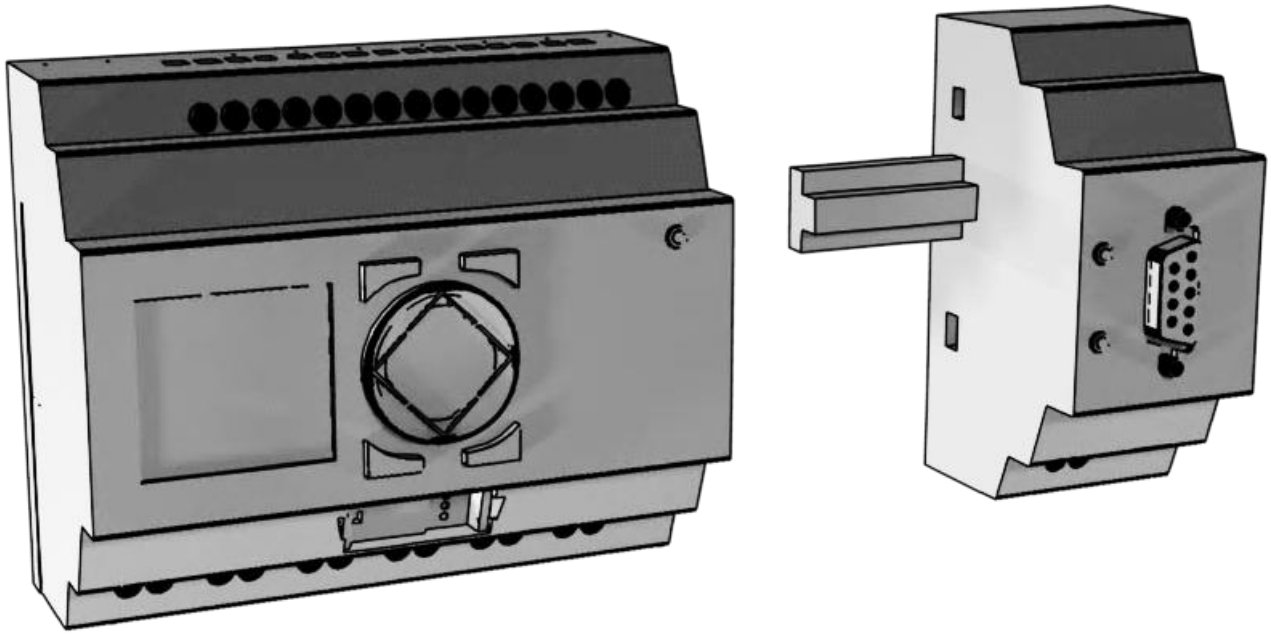


Figure 154: EASY719 and EASY204-DP preset

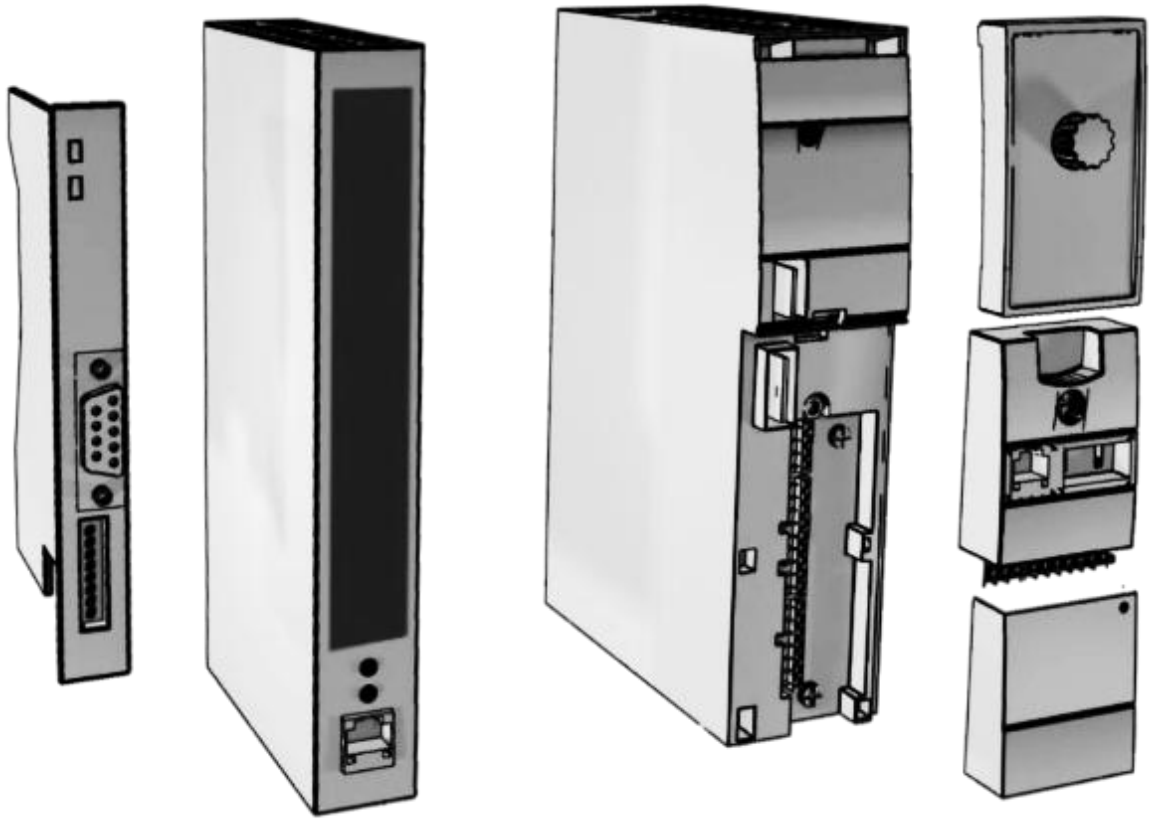


Figure 155: MOVITRAC B and DFP21B preset

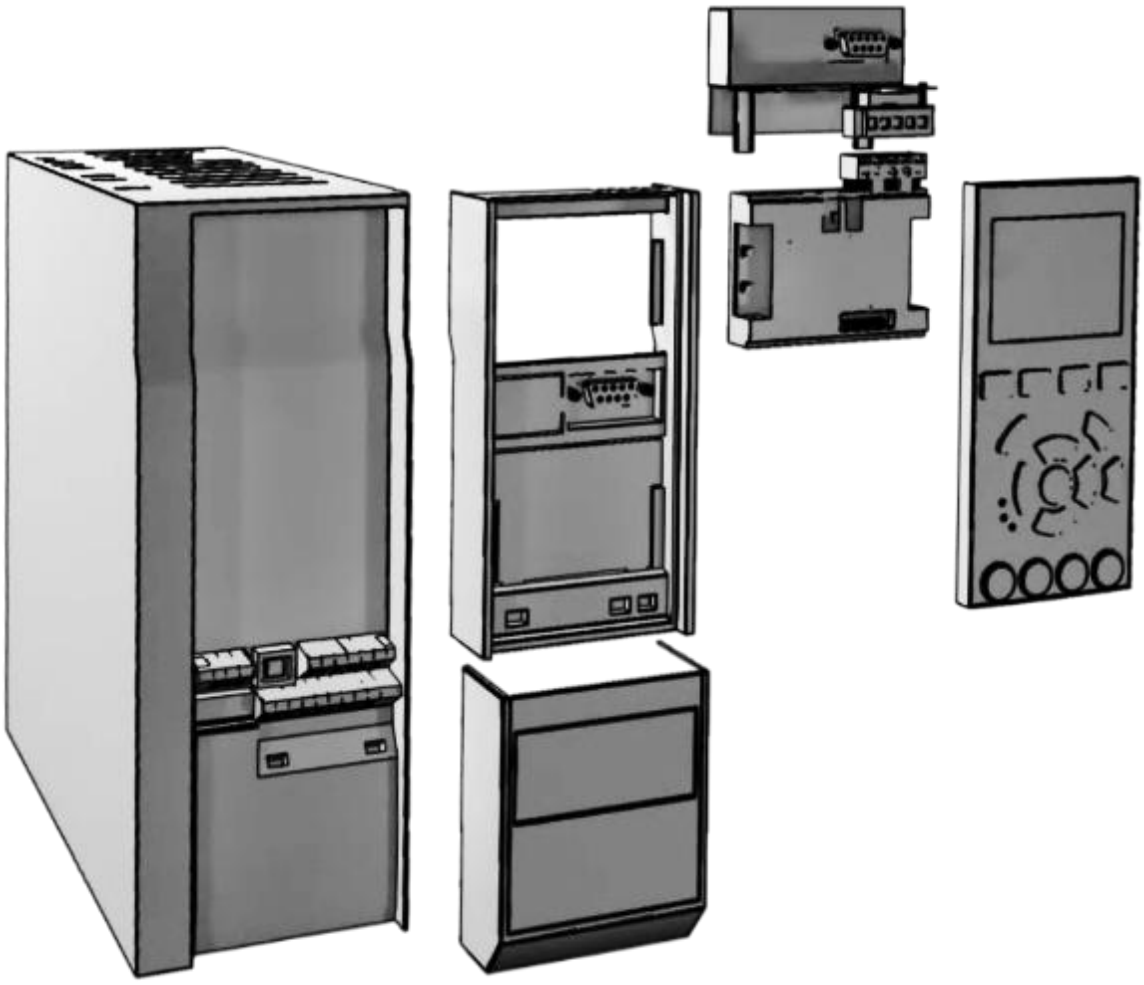


Figure 156: HVAC FC102 and MCA101 preset

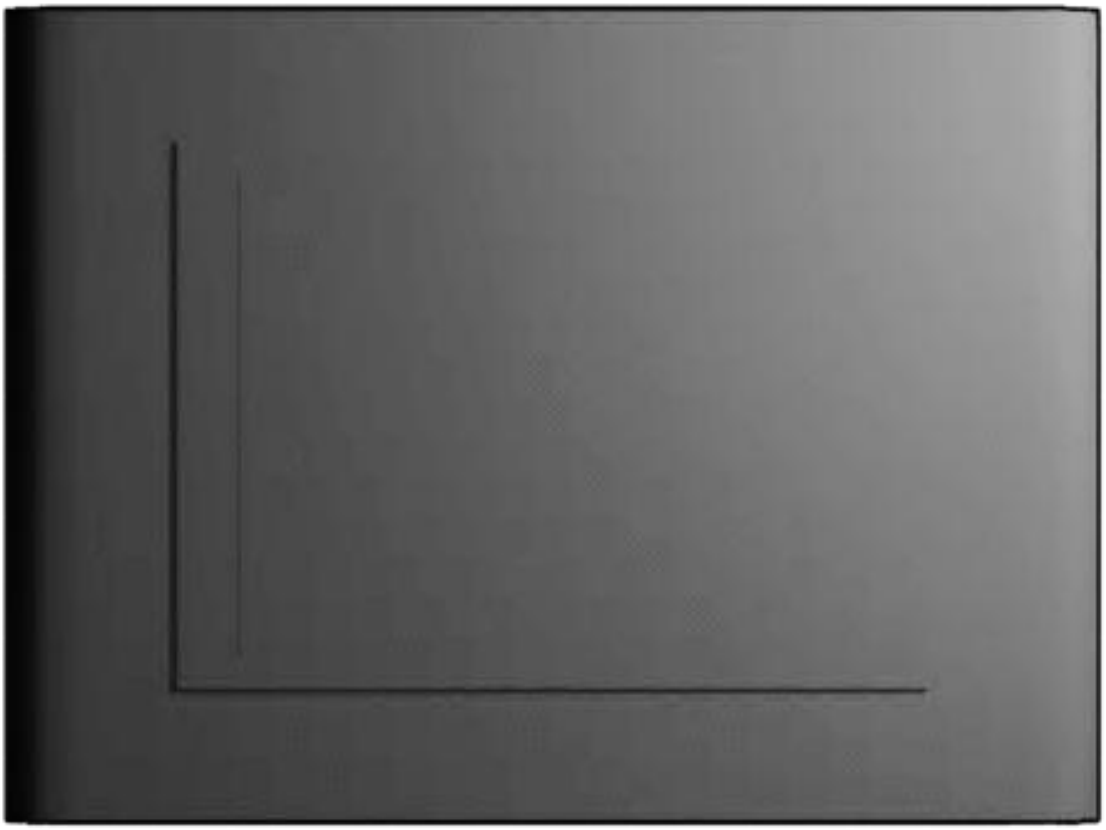


Figure 157: TP 177B 6" preset

Figure 158: RS485 repeater preset

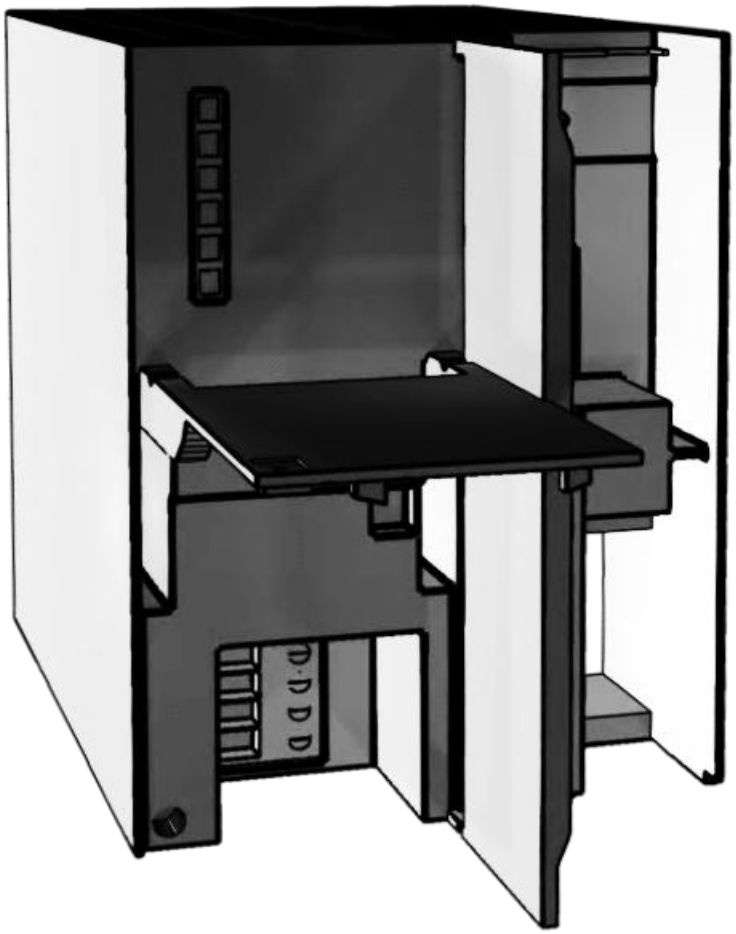


Figure 159: DP/PA coupler preset

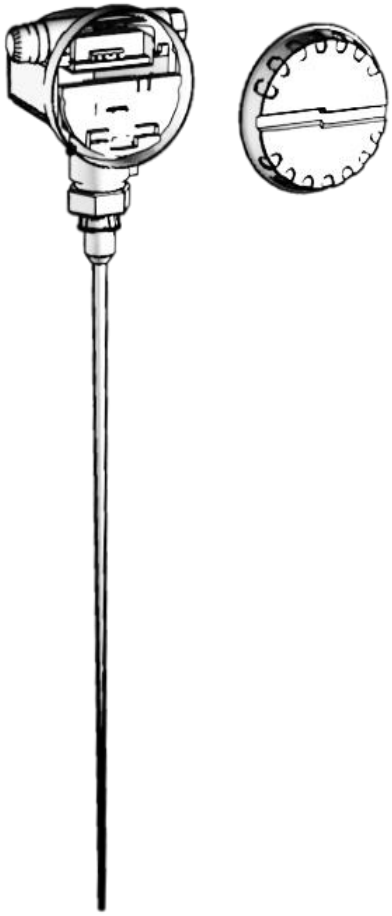


Figure 160: Levelflex M FMP40 preset

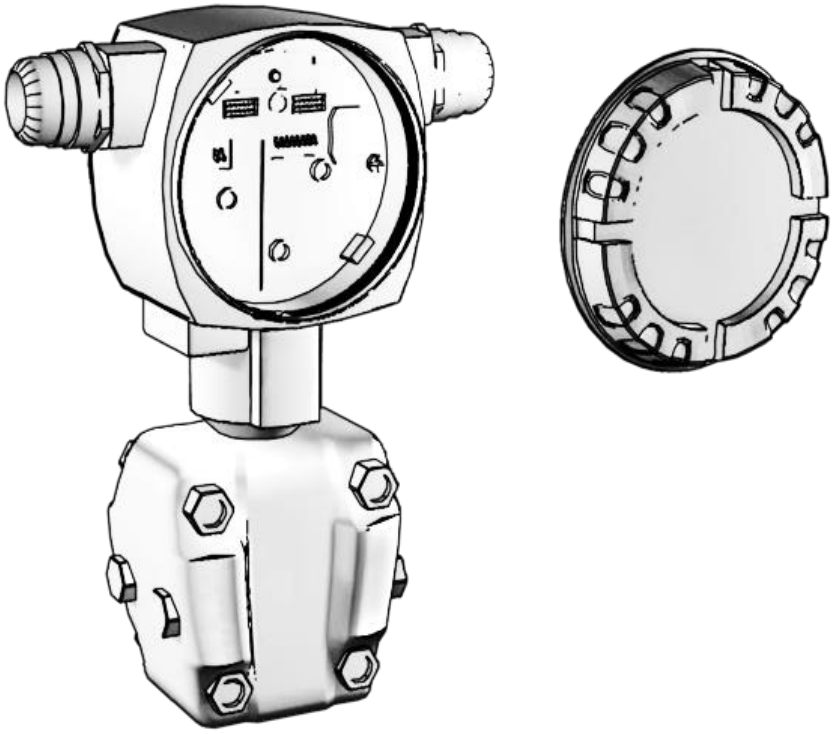


Figure 161: Deltabar S PMD70 preset

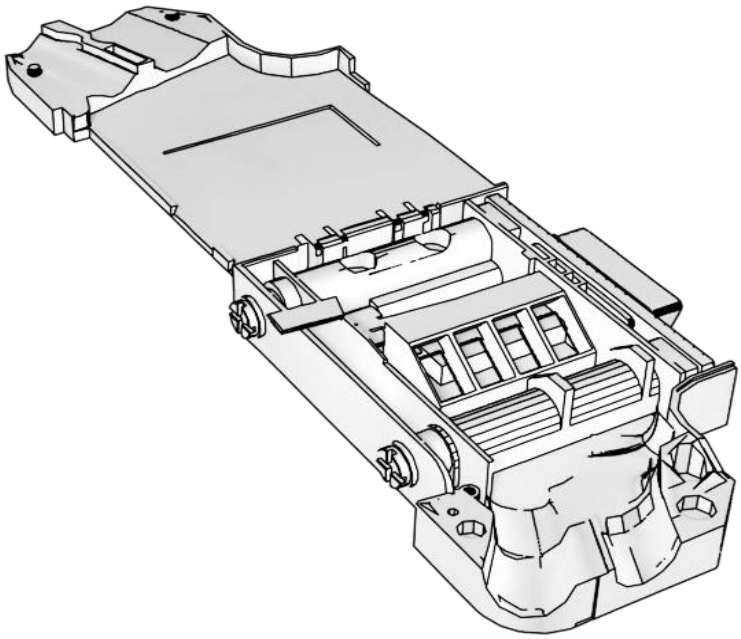


Figure 162: Profibus DP connector preset

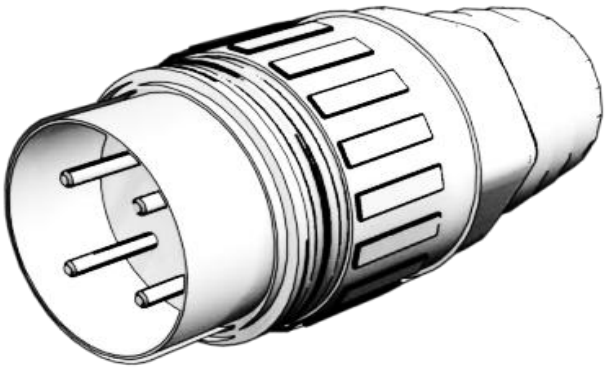


Figure 163: Profibus PA connector preset

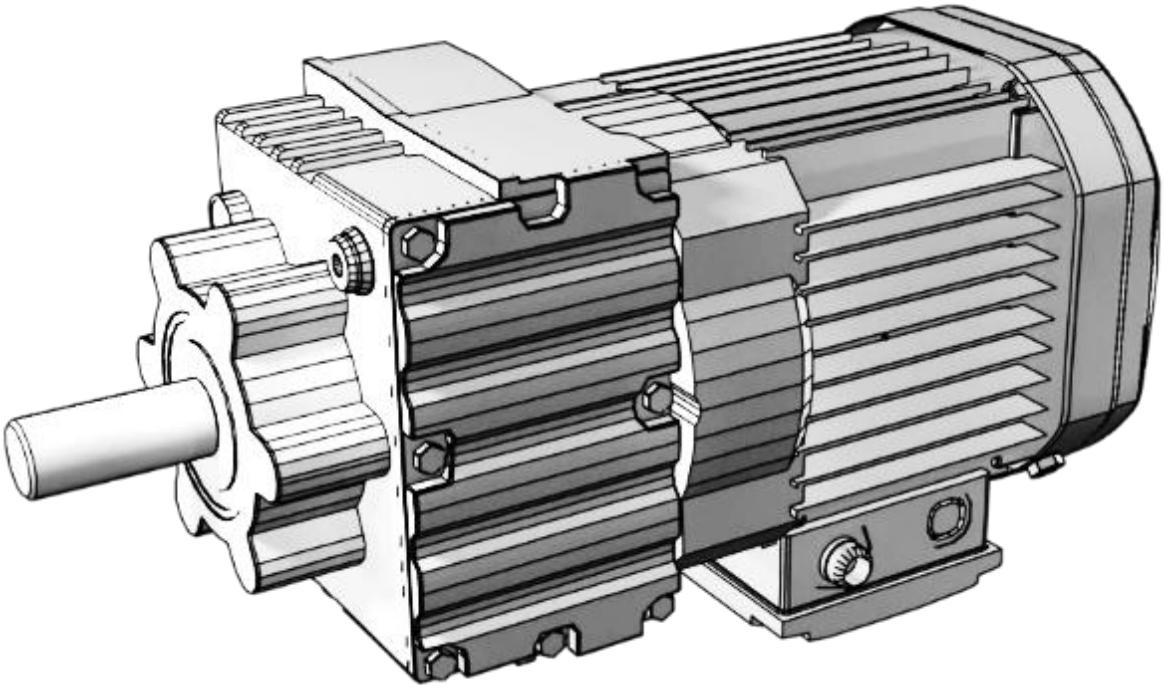


Figure 164: Single phase motor preset

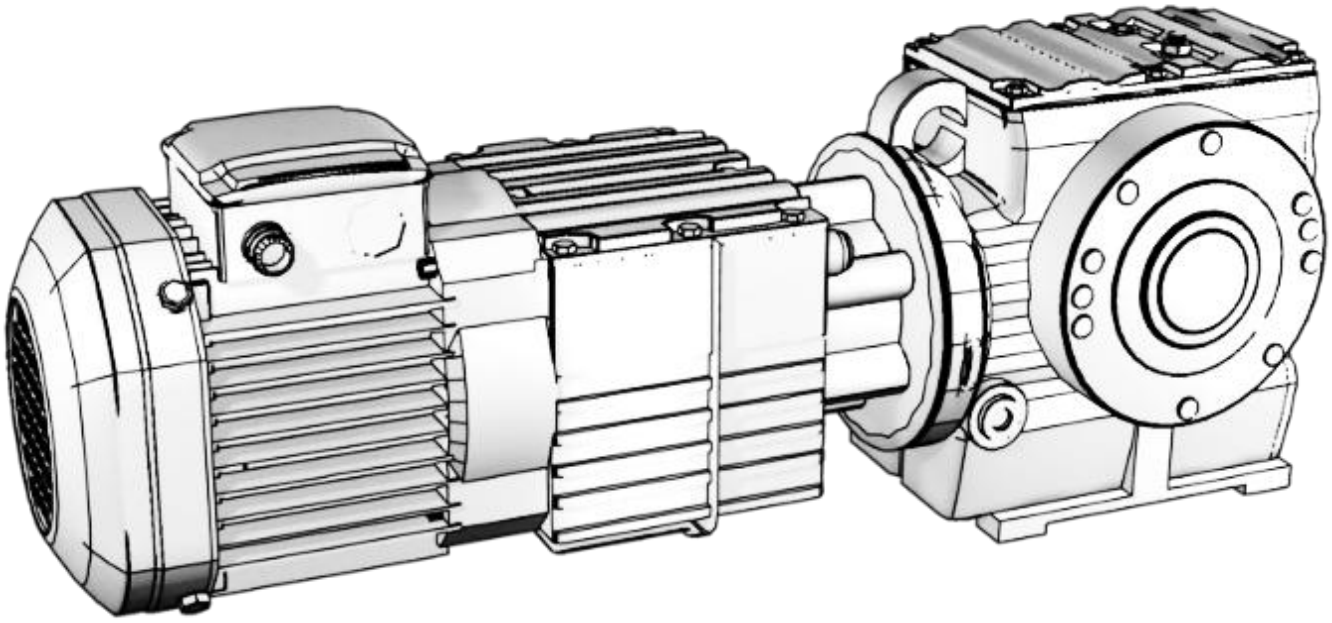


Figure 165: 3 phase motor preset

This is the end of Appendix IV.

Appendix V to Appendix XIII Configuration Instructions (page 213 to page 682) were separated from the thesis report as standalone documents. They were embedded in the portfolio document where the thesis report is.