

Ali Gulabi

Development of an Embedded SCADA System with
PLC and Java Application for Synchronous Operation
of Standard Servo Drives

Master thesis based on the examination and study regulations for the
Master of Engineering degree programme
Information Engineering
at the Department of Information and Electrical Engineering
of the Faculty of Engineering and Computer Science
of the University of Applied Sciences Hamburg

Supervising examiner: Prof. Dr. Ing Gustav Vaupel
Second examiner: Prof. Dr. Ing Reinhard Müller

Day of delivery May 21st 2007

Ali Gulabi

Title of the Master Thesis

Development of an Embedded SCADA System with PLC and Java Application for Synchronous Operation of Standard Servo Drives

Keywords

SCADA, S7, PLC, Simovort Masterdrives MC, Java Applets, S7 Java beans, Parameterization, Servo Drives, SIMATIC S7-300, CP 343-1 IT, PROFIBUS DP, MPI, CP5613 A2.

Abstract

In this project an embedded Supervisory Control and Data Acquisition system is developed for synchronous operation of standard servo drives by using PLC, CP 343-1 IT, Simovort Masterdrives MC, Servo drivers, a PC, and Communication Processor CP5613 A2. The components are networked via PROFIBUS DP, and Ethernet. The system can be accessed and controlled through Internet or Ethernet. The Human Machine Interface (HMI) is designed by using Siemens S7 Java beans and Java swing components, and then it is downloaded to CP 343-1 IT via FTP protocol; where the CP343-1 IT acts as a server. There is also an S7 program designed to control the Masterdrivers MCs, and to manage the data flow between HMI and drivers.

Ali Gulabi

Thema der Masterarbeit

Entwicklung eines eingebetteten SCADA Systems mit dem PLC und der Java-Anwendung für Gleichlaufbetrieb der Standardservo-Antriebe

Stichworte

SCADA, S7, PLC, Simovort Masterdrives MC, Java Applets, S7 Java beans, Parametrierung, Servoregler, SIMATIC S7-300, CP 343-1 IT, PROFIBUS DP, MPI, CP5613 A2.

Kurzzusammenfassung

In diesem Projekt wurde für den Gleichlaufbetrieb eines Standardservo-Antriebes ein Embedded Überwachungssteuer- und Datenerfassungssystem entwickelt. Das System wurde mit folgenden Komponenten entwickelt:

- PLC
- CP 343-1 IT
- Simovort Materdrives MC
- PC- und Kommunikation Prozessor CP5613 A2

Man kann auf das System via Internet/Ethernet zugreifen, steuern und Daten erfassen. Die Benutzeroberfläche wurde mit Siemens S7 Java beans und Java Swing entwickelt und über ftp zu dem Kommunikation Prozessor CP 343-1 IT übertragen. Der CP 343-1 IT ist ein Server. Die Kommunikation, Steuerung und Datenerfassung zwischen GUI und dem Masterdrives MC geschieht über S7 Programm

1 Acknowledgement

I would like to thank to the people who contributed so much during course of my master thesis. It was a very level and professional environment, which not only let me develop technical skills, but also help me to improve academic research methodology.

Prof. Dr.-Ing. Gustav Vaupel; I thank you for offering the project, and taking your time to supervise it. You were very instructive and helpful.

Prof. Dr.-Ing. Reinhard Müller; I thank you for taking part in my thesis to be the second supervisor.

Dipl.-Ing. Frank Korpel; our cooperation and willing to help were exceptional. You were available almost every time I needed help. Thank you very much.

ALI GULABI

University of Applied Sciences Hamburg

May 21st 2007

1	<u>ACKNOWLEDGEMENT</u>	3
2	<u>PREFACE</u>	7
3	<u>INTRODUCTION</u>	9
3.1	CHAPTER OVERVIEW	9
4	<u>SYSTEM</u>	11
4.1	HARDWARE	11
4.1.1	SIMOVERT MASTERDRIVES MC SETUP 6SX-7000-0AF10	11
4.1.1.1	Masterdrives MC 6SE7012-5EP50-Z	11
4.1.1.2	Masterdrives MC 6SE7011-0TP50-Z (Right)	11
4.1.2	MOTORS:	12
4.1.2.1	SIEMENS 3-Permanent-Magnet-Motor 1FK6032-6AK71-1TG0	12
4.1.2.2	SIEMENS 2-Magnet-Motor 1FT6031-4AK7	12
4.1.3	SIEMENS SIMATIC S7-300 SETUP	12
4.1.4	COMMUNICATION BOARD, PC AND CABLES	13
4.2	SOFTWARE	13
5	<u>ASSEMBLY</u>	14
5.1	CP5613 A2 COMMUNICATIONS PROCESSOR	14
5.1.1	INSTRUCTIONS FOR ASSEMBLY	14
5.2	PROFIBUS	15
5.2.1	SPECIFICATIONS	16
5.2.2	BUS CONNECTION FOR THE FIRST AND LAST NODE ON PROFIBUS	17
5.2.3	INSTALLING COMPONENTS ON THE RAIL	19
5.2.4	POWER SUPPLIES	20
6	<u>INDUSTRIAL COMMUNICATIONS</u>	21
6.1	THE MULTIPOINT- INTERFACE (MPI)	23
6.2	THE PROFIBUS DP	25
6.3	ETHERNET	26
7	<u>MOTION CONTROLLERS</u>	29
7.1	PARAMETERIZATION	31
7.1.1	PARAMETERIZATION VIA PMU	35
7.1.2	PARAMETERIZATION VIA OP1S (6SE7090-0XX84-2FK0)	35
7.1.3	PARAMETERIZATION VIA DRIVE MONITOR (STAND ALONE)	38
7.1.3.1	Install Drive Monitor	38
7.1.3.2	Parameterize the slave (Masterdrives MC)	39
7.1.4	PARAMETERIZATION VIA DRIVE MONITOR (SIMATIC MANAGER)	41
7.2	PARAMETERIZATION STEPS	43
7.2.1	MAIN STEPS FOR PARAMETERIZATION	44

7.2.1.1	Power section definition (P060 = 8)	44
7.2.1.2	Board Configuration (P060 = 4)	46
7.2.1.3	Drive Setting (P060 = 5)	47
7.2.1.4	Function adjustment	48
8	<u>MODERN CONTROL</u>	49
8.1	SYNCHRONOUS OPERATION	49
8.1.1	BASIC POSITIONING	49
8.1.2	TECHNOLOGY OPTION F01	51
8.1.3	GENERAL FUNCTIONS OF TECHNOLOGY OPTION	51
8.1.4	POSITIONING	53
8.1.5	SYNCHRONIZATION	53
8.2	APPLICATION	54
8.2.1	OPERATIONS MODES	56
8.2.1.1	Synchronous Operation	56
8.2.1.2	Reference run (MDI = 0)	56
8.2.1.3	Reference run (MDI = 1)	57
9	<u>COMMUNICATION WITH THE MASTERDRIVES MC</u>	58
9.1	FUNCTIONALITY OF THE CBP	59
9.2	USEFUL DATA	59
9.3	STARTING THE DP SLAVE THROUGH S7-PROJECT	64
10	<u>S7- PROGRAMMING AND CONTROL</u>	67
10.1	SYNCHRONOUS OPERATIONS	70
10.1.1	PARAMETERIZATION	70
10.1.2	FB10 CONVERSION	75
10.1.3	FC2 WRITE & READ OPERATIONS FOR CONTROL AND STATUS WORDS	77
10.1.4	FB60 USER FRIENDLY HMI	80
10.2	ASYNCHRONOUS OPERATION	81
10.3	RESET FUNCTIONALITY	82
11	<u>HUMAN MACHINE INTERFACE (HMI)</u>	83
11.1	ECLIPSE	84
11.1.1	THE ECLIPSE'S LICENCE	84
11.1.2	HOW START USING ECLIPSE	85
11.1.3	DOWNLOAD THE JAVA S7 BEANS LIBRARY FROM SIEMENS SITE	85
11.1.4	JAVA VISUAL PROJECT	85
11.2	S7-BEANS AND INTERCONNECTION HIERARCHY	88
11.2.1	S7-BEANS	89
11.2.2	HIERARCHICAL RELATIONSHIP BETWEEN COMPONENTS	89
11.3	JAVA APPLET FOR DISPLAYING THE HMI ON A WEB BROWSER	92
11.4	DEVELOPING HMI IN JAVA AND CONNECTING IT WITH S7 PROGRAM	94
11.4.1	SIEMENS S7 BEANS	94
11.4.1.1	Siemens S7 Device beans	94
11.4.1.2	Siemens S7 GUI beans	95
11.4.1.3	Siemens S7 Utility beans	96
11.4.2	HMI DESIGN PAGE 1	96
11.4.2.1	Modifications to the class	97

11.4.2.2	Inserting S7 Device Beans	98
11.4.2.3	Design GUI	101
11.4.3	HMI DESIGN PAGE 2 (SYNCHRONOUS OPERATION)	105
11.4.3.1	Adding Tacho	105
11.4.3.2	Adding text box for displaying the speed	107
11.4.3.3	Adding text box for setting the speed	108
11.4.3.4	Adding Digital Input (DI) buttons	109
11.4.3.5	Adding navigation buttons	110
11.4.4	USER FRIENDLY HMI DESIGN SYNCHRONOUS OPERATION	111
11.4.5	HMI DESIGN FOR ASYNCHRONOUS OPERATION	113
12	<u>CONCLUSION</u>	115
12.1	SUGGESTIONS FOR THE FUTURE WORK	116
13	<u>REFERENCES</u>	117
	<u>APPENDIX</u>	118
A	<u>CONFIGURATION</u>	118
A.1	PC	118
A.2	S7-HARDWARE CONFIGURATION	119
A.2.1	HARDWARE CONFIGURATION	122
A.2.2	ADDING SIEMENS SIMOVERT MASTERDRIVES MOTION CONTROLLERS	128
B	<u>FUNCTION DIAGRAMS</u>	131
C	<u>INSTRUCTIONS TO START HMI FROM THE INTERNET</u>	135
D	<u>CD-ROM</u>	140
E	<u>LIST OF FIGURES</u>	141
F	<u>LIST OF TABLES</u>	143

2 Preface

In the past industrial companies used separate networks or network cells such field level and cell level networks in production and other networks for the process level. Following the development in information technology all these smaller networks linked to each other.

Especially in manufacturing companies an automaton network concept developed under the name of Totally Integrated Automation (TIA). TIA includes actuator sensor level, field level, cell level and process control level, which makes use of actuator-sensor interface, PROFIBUS, PROFIBUS and industrial Ethernet, and industrial Ethernet respectively. Through TIA it is possible to view or control all the levels all the way to the actuators from process control level. In recent years many companies started opening divisions in many countries around the world, and wanted to connect and control any devising from another or any place around the world (Distributed Management).

Distributed management can be realized through Supervisory Control and Data Acquisition (SCADA) system. It is a common process control application that collects data from sensors on the shop floor or in remote locations and sends them to a central computer for management and control. A SCADA system includes input/output signal hardware, controllers, Human Machine Interface (HMI), networks, communication, database and software.

The term SCADA usually refers to a system with a central unit that monitors and controls a complete site or a system spread out over a long distance. The bulk of the site control is actually performed automatically by a Remote Terminal Unit (RTU) or by a Programmable Logic Controller (PLC). The access rights for the host and its control functions are almost always restricted to basic site over-ride or supervisory level capability.

This concept can also be used in power plants as well as in oil and gas refining, telecommunications, transportation, and water and waste control, and also being used in chambers that are hazardous to human health or difficult to access

In the past the Internet was very slow and unreliable, nowadays reliability is increased and speed limit been pushed towards real-time. For example expansion in Internet telephone usage is due to increase in quality of service of the Internet. Therefore it is understandable to see similar usage of the Internet in industrial and automation areas as well. Today there are many companies that are already connect their branches around the world, and it is understandable to connect production floors as well.

The goal of this project is to design an embedded SCADA system including an HMI (Human Machine Interface), embedded in a web page through applets, which visualizes and controls a standard servo drive in a remote area.

The HMI will be developed by using an open source java editor (Eclipse) and Siemens java APIs to program in java. The HMI is machine and location independent, can be accessed through internet anywhere in the world.

There will be a Siemens IT-343-1 device and a Siemens SIMATIC S7-300 CPU in between HMI and Simovert Masterdrives MCs. IT-343-1 device should be configured, and S7-300 unit must be programmed such a way so the user can communicate and control the Motors from the Internet. Simovert Masterdrives MCs must be programmed and parameterized in order to be communicated by S7-300 unit.

3 Introduction

This document is written and organized such a way that it can be used in chronicle order or any given chapter can be used independently. It is assumed that the reader has some background in control theory and automation, and in information engineering. It is not scope of this documentation and also not possible to explain all the details of all the relevant topics. Therefore only project specific information is covered carefully; otherwise documentation would be massive and go out of control. For example Compendium for Simovert Masterdrives MC V1.66 is about 1500 pages which include very important information about Siemens Masterdrives. If the user is not satisfied with the information for any given chapter, he or she can take advantage of the references.

3.1 Chapter overview

There are six chapters and an appendix in this document:

- Chapter 1: Acknowledgement
- Chapter 2: Abstract
- Chapter 3: introduction
- Chapter 4: Modules and the components that are used in this system are listed.
- Chapter 5: The assembly of the communication processor CP5613 2A, and PROFIBUS is given.
- Chapter 6: The industrial communications briefly explained, and then means of communications that are used in this project are explained.
- Chapter 7: Motion controllers and parameterization methods of them is given. This is very essential for accessing Masterdrives MCs.
- Chapter 8: In this chapter operation modes, and the modern control principals behind them is given. Digital Inputs (DI) and DI assignment also described.
- Chapter 9: Communications board PROFIBUS (CBP) is used to link Masterdrives MCs (slaves) to the PROFIBUS bus system, therefore; the CPB and its telegram types are examined and explained.
- Chapter 10: An S7 control program is developed to implement the functionalities of the HMI
- Chapter 11: HMI development using Java programming language and siemens S7 beans in Eclipse environment.
- Appendix 1: Configuration of PC and SIMATIK S7-300 are explained in this section.

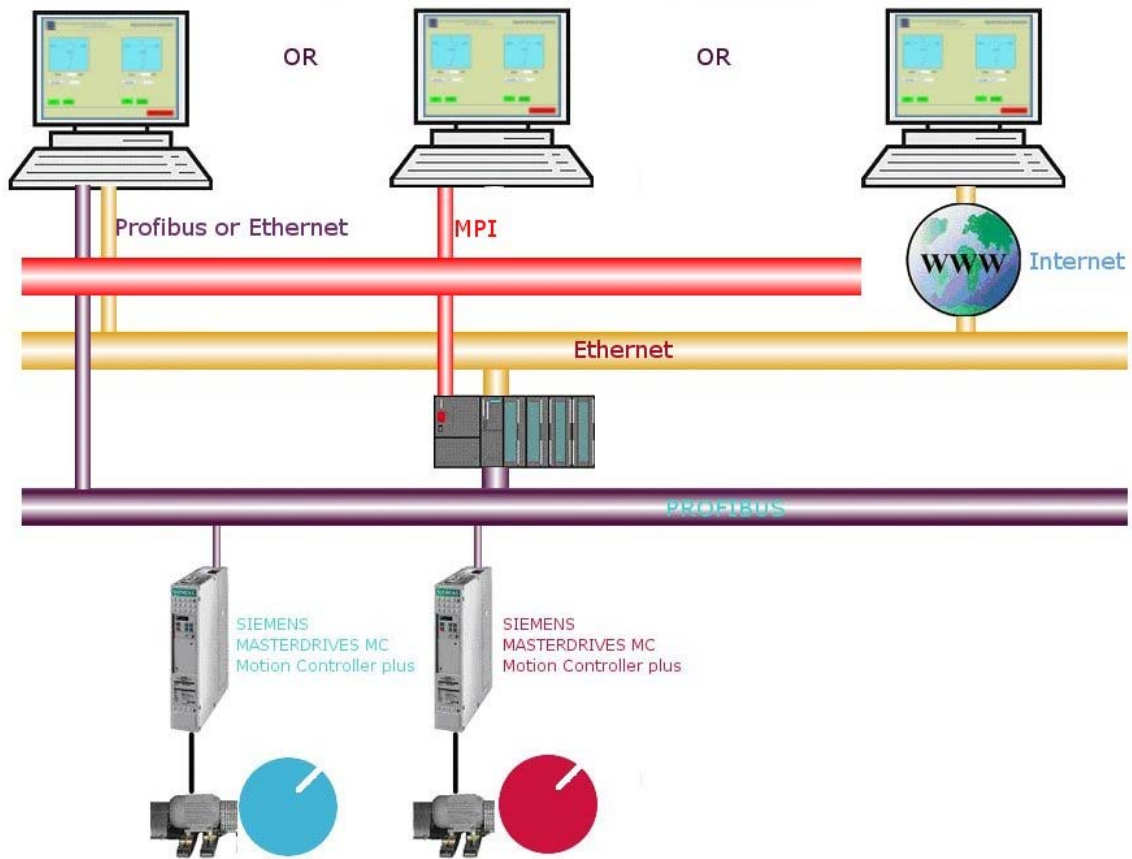


Figure 3.1The system description

Here is the model of finished system. It can be accessed via MPI, PROFIBUS or Ethernet, or via the Internet. MPI is mostly used for local access for diagnostic reasons or system updates. PROFIBUS and Ethernet is mainly used companywide, but internet access can be used anywhere in the world. The Internet access enables the distributed management Supervisory Control and Data Acquisition (SCADA).

4 System

4.1 Hardware

4.1.1 SIMOVERT MASTERDRIVES MC Setup 6SX-7000-0AF10

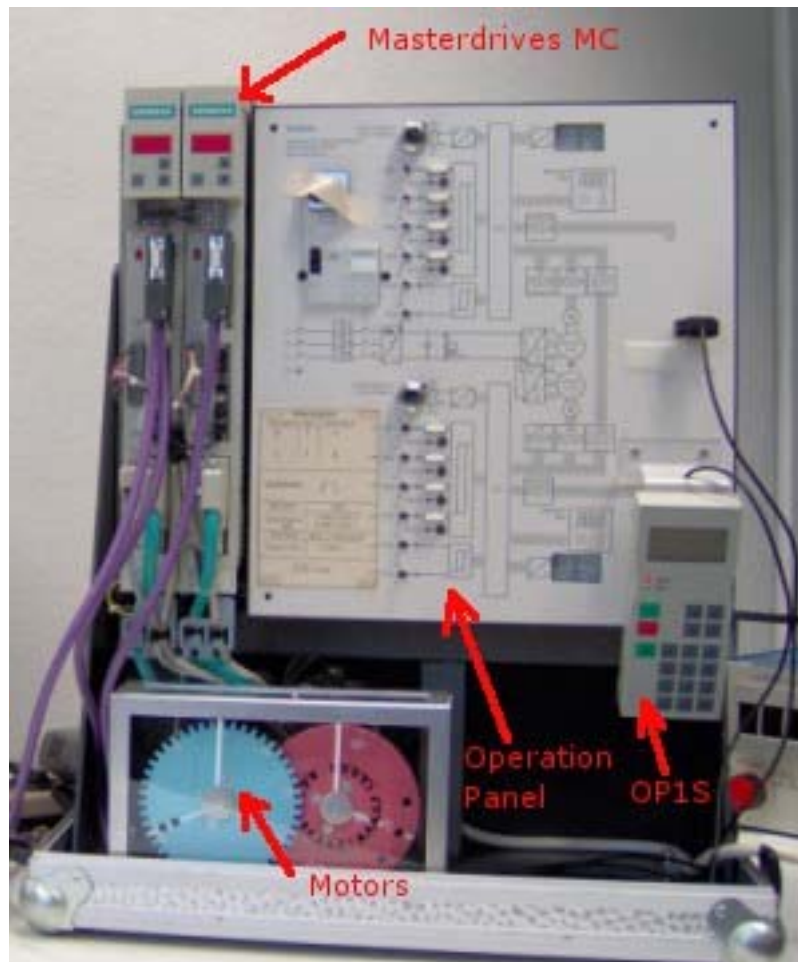


Figure 4.1 SIMOVERT MASTERDRIVES MC

4.1.1.1 Masterdrives MC 6SE7012-5EP50-Z

V_{in} 3AC 380-480V 1.7A

V_{out} 3AC 0-380..480V

I_{out} 1.5A

4.1.1.2 Masterdrives MC 6SE7011-0TP50-Z (Right)

V_{in} DC 510-650V

V_{out} 3AC 0-380..480V

I_{out} 2A

4.1.2 Motors:

4.1.2.1 SIEMENS 3-Permanent-Magnet-Motor 1FK6032-6AK71-1TG0

Nr.E K883 7887 01 007
MO/N 1.1/0.8 Nm Io 1.7A UIN 252V
nN/max 6000/8300 min⁻¹ EN 60034
OPIS 6SE7090-0XX84-2FK0

4.1.2.2 SIEMENS 2-Magnet-Motor 1FT6031-4AK7

There is no information sticker on the second motor. Since this training unit was configured by siemens; the original parameter in Masterdrives MC must reflect the parameters of the motor. Here is the motor information taken from Masterdrives MC by using Drive Monitor:

Motor rotor speed:	6000 min ⁻¹
Motor rotor frequency:	200 Hz
Max power factor:	0.800
Motor rotor torque:	0.75 Nm

4.1.3 Siemens SIMATIC S7-300 Setup

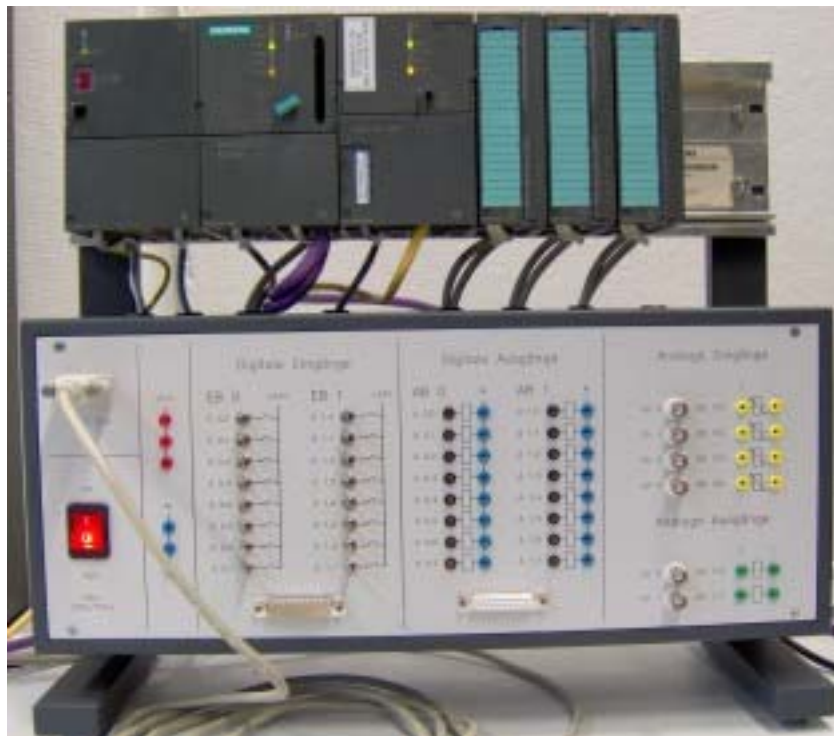


Figure 4.2 SIMATIC S7-300

- Siemens SIMATIC S7-300 CPU
CPU315-2DP 315-2AF03-0AB0
- SIMATIC NET
CP 343-1 IT 343-1GX20-0XE0
- DC 24V Power Supply
PS307 5A 307-1EA00-0AA0
- Digital Input
SM321 321-1BH02-0AA0
DI 16xDC24V
- Digital Output
SM322 321-1BH01-0AA0
DO 16xDC24V/0.5A
- Analogue Input/Output
SM334 334-0KE00-0AB0
AI4/AO2x12BIT

4.1.4 Communication Board, PC and cables

- CP5613 Communication Board
- PROFIBUS cable and Connectors
- Personal Computer (PC)
- RJ45
- PG cable

4.2 Software

- Drive Monitor V5.3.2.2
- Drive ES V5.3 + SP3
- SIMATIC Manager
STEP 7 Version V5.3 + HF2
- Eclipse Version 3.2
- Java 2 Runtime Environment, SE V1.3.2
- J2SE Runtime Environment 5.0
- Siemens API for Java
Release V2.5.5

5 Assembly

5.1 CP5613 A2 Communications Processor

In configuration part, we will see that CP5613 A2 communications processor is capable of communicating thorough PROFIBUS DP, and MPI. In order to set PROFIBUS up one must use MPI to download the system configuration to the CPU, but this part will be explained in configuration part in detail.



Figure 5.1 CP5613 A2

1. The CP 5613 A2 is a PCI communications card for connecting PCs to PROFIBUS with Windows 2000, Windows XP and Windows Server 2003.
2. CP 5613 A2 optimized for the fast DP master mode, which can handle up to 124 DP slaves.
3. Plug-and-Play support
4. Floating RS-485 attachment

5.1.1 Instructions for assembly

1. install the product software, as follows:

- After installation there will be an icon



PG-PC-Schnittstelle einstellen.Ink

2. Install the CP 5613 A2 module. These instructions are same standard instructions for installing an electronic device or a PCI card to PC;
 - Make sure to discharge static electric from your body, from computer and the tools.
 - Do not touch the CP 5613 A2 communications processor from pins or conductors.
 - PC must be turned off during installation.
 - Insert the CP 5613 A2 to an available PCI slot

5.2 PROFIBUS

PROFIBUS bus connector

RS 485-IS; 6ES7 972-0BA50-0XA0

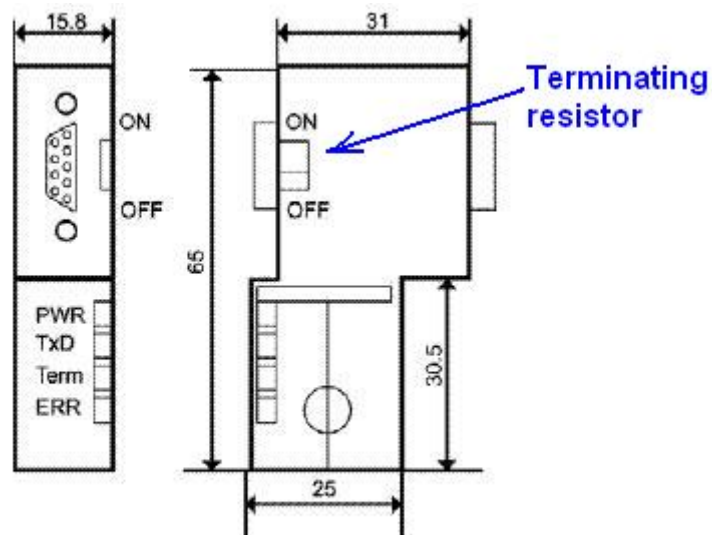


Figure 5.2 PROFIBUS connector

5.2.1 Specifications

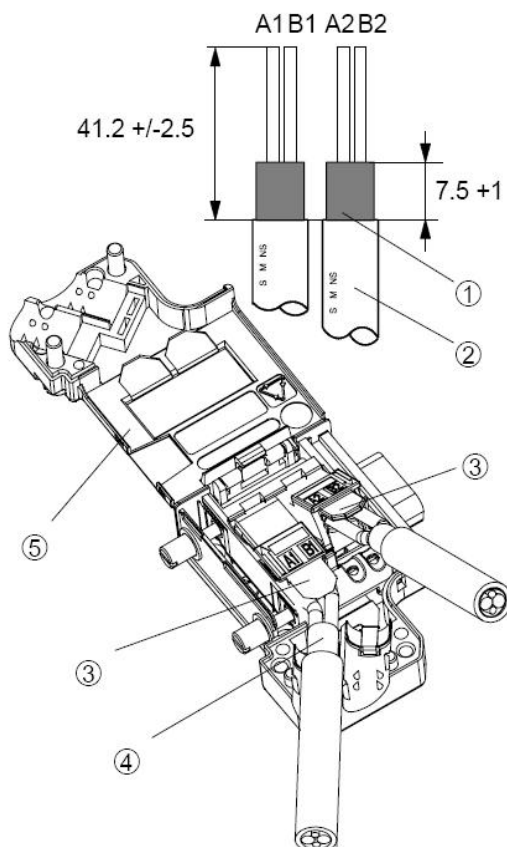
Mechanical

PROFIBUS	9 Pole SubD Pin Headers
Programming/Diagnostics	9 Pole SubD Socket
Insertion (withdrawal) cycles	Min. 200
Cable Type	Solid Core PROFIBUS Type A, EN50170
Cable Diameter	8mm
Screw/Tightening Torque	4-40 UNC/0.4Nm
Enclosure Material	Die-Cast Zinc
Temperature Range	-20°C to +75°C
Cable Connection	IDC Technology
Terminating Resistor	Yes, Built-In Switchable
Bus Signals	Dual, IN and OUT

Insulation Stripping Lengths

Outer Sheath	17mm
Shield	11mm

Wires have to be stripped as shown



Bus cable installation

- (1) Cable shield
- (2) Bus cable (e.g. 6ES7 972-0BA50-0XA0)
- strip insulation, e.g. with stripping tool 6GK1905-6AA00
- (3) Contact cover for insulation-piercing connecting device
- insert the green and red cores into the open contact cover right up to the end
- close the contact cover (press it down as tight as possible)
- (4) Insert the cable into the opening (cable shield must lie bare on the metal guide)
- (5) Close the housing cover and screw it shut

[1]

5.2.2 Bus connection for the first and last node on PROFIBUS

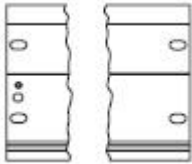




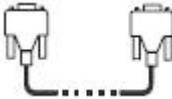

Cable must always be connected on the left (see label A1, B1)
Switch position **must** be "ON" for the first and last node on the PROFIBUS, where terminating resistance is connected, which means the bus line is terminated.



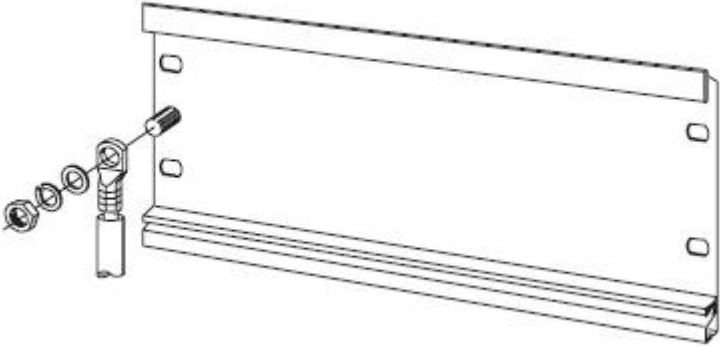
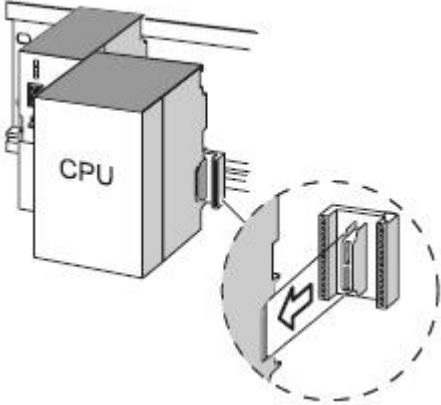
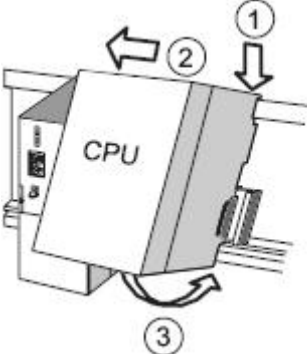
Figure 5.3 PROFIBUS connections

S7-300

Here are some of the main components of S7-300 module

Component	Function	Picture
Mounting rail	S7-300 racks	
Power supply PS307, 5A	To convert line voltage (123/130 VAC) 24 VDC	
CPU CPU315-2 DP	To evaluate the user program and communicate with the peripherals	
Communication Processor CP 343-1 IT	To connect to Ethernet/Internet	
Signal modules(SM) Digital Input (DI) SM321	It matches different process signal levels to the S7-300 16xDC24V	
Digital Input (DO) SM322	16xDC24V,0.5A	
Analogue Input/output SM334	AI4/AO2x12BIT	
PG cable	Connects PG/PC to the CPU	
RJ45	To connect to internet	

5.2.3 Installing components on the rail

1	Connecting the protective conductor	
2	Mounting order	<ol style="list-style-type: none"> 1. Power supply module 2. CPU 3. Communication Processor 4. SMs, IMs
3	<ul style="list-style-type: none"> - Plug the bus connectors into the CPU and SMs/CPs - Except for the CPU, each module is supplied with a bus connector. - Always start at the CPU when you plug in the bus connectors. Remove the bus connector from the "last" module of the assembly. - Plug the bus connectors into the other modules. The "last" module does not receive a bus connector. 	
4	Mount all modules to the rail in the order given at step 2, slide them up to the module on the left, and then swing them down.	

5.2.4 Power Supplies

- Masterdrives MC: 380/220 V, 50Hz, 63A
- S7-300 Unit: 220V, 50Hz, 16A



Figure 5.4 Power supply 380 V



Figure 5.5 Power supply 220 V

6 INDUSTRIAL COMMUNICATIONS

It is not focus of this document to go in detail of all types of industrial communications, instead, as it is seen from the Figure 6.1 to explain the types of communications that have been used during development of this project.

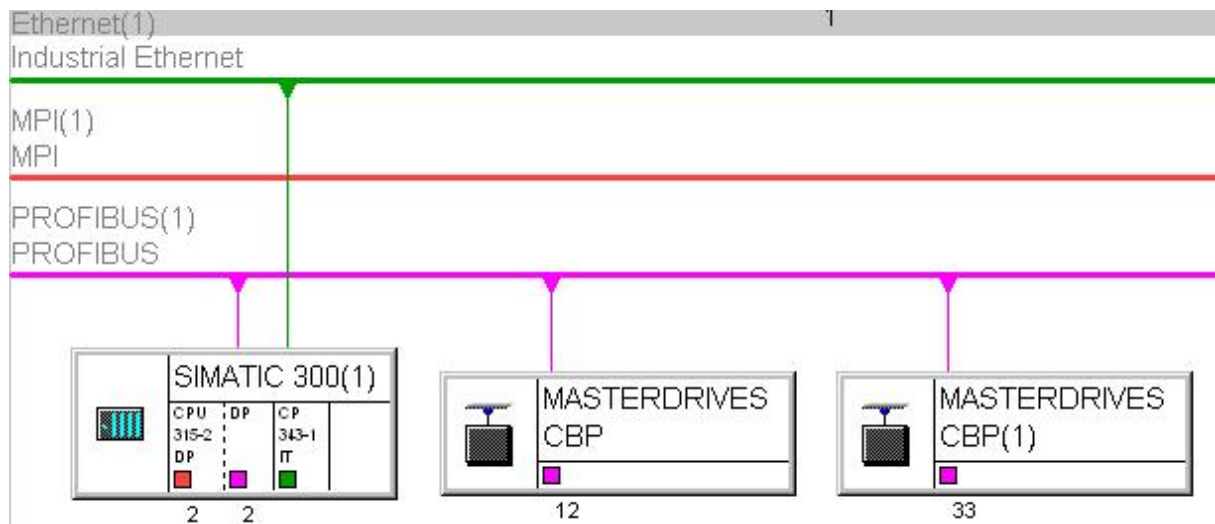


Figure 6.1 A screen shot of S7 communication module

Communications networks play very important role in the modern automation solutions. They allow to interconnect components and devices to control and exchange information between different automation levels.

Industrial networks must comply with certain norms and meet high quality requirements over and above those of normal networks to have successful automation system and production plant.

What is the network used for?

- Link I/O to the controller?
- Link PLCs and operator interfaces together?
- Link computers in manufacturing together?
- Connect manufacturing with the rest of the company?
- Link manufacturing with other plants e.g. suppliers

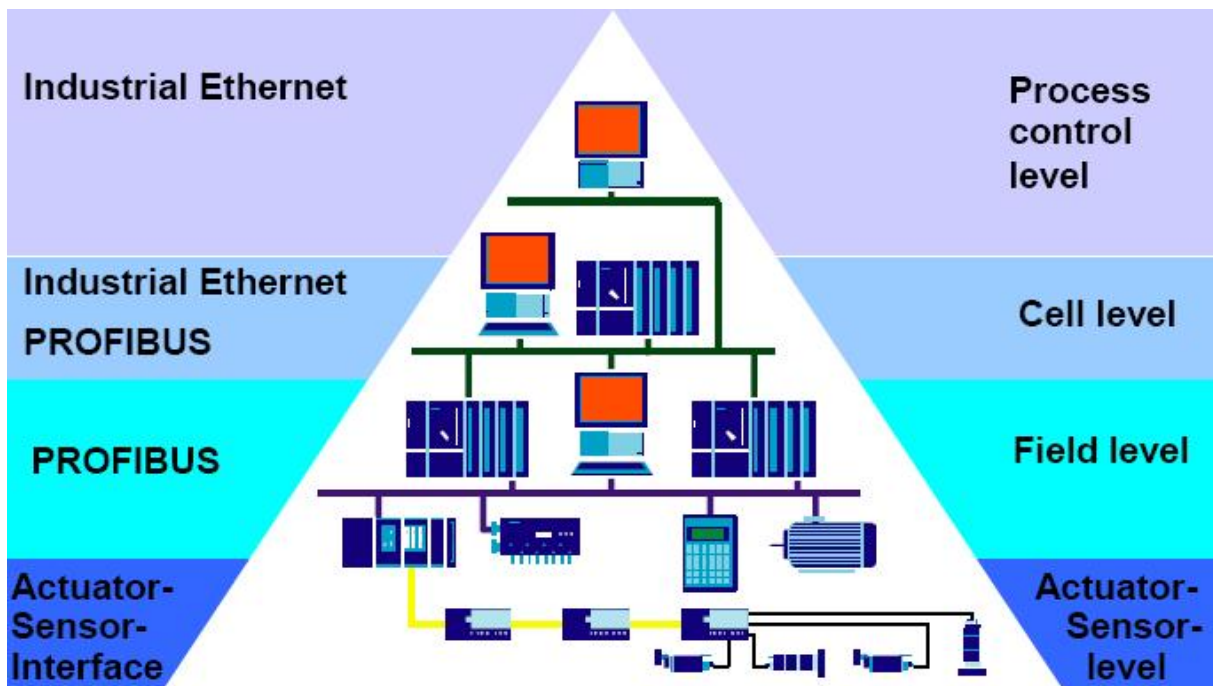


Figure 6.2 Industrial communication pyramid [2]

As it is seen in the communication pyramid industrial communications system must allow the connection of the simplest sensors and actuators to controllers and the connection of controllers to each other and to computers.

Certain factors in industrial environment such as electromagnetic interference, high levels of chemical contamination, dampness or mechanical strain puts high demands on the network structure and network components.

Industrial communications networks belong to the LAN group, even though by internetworking LANs with Internet. It is used worldwide transmission of selected information from the production area to another is nevertheless possible. Also it enables to control and visualize plants in remote areas.

Figure 6.3 shows the methods to communicate with Masterdriver MCs using PC (Drive Monitor) or OP1S. During this project MPI is only used during hardware configuration, between SIMATIC S7-300 and PC. Serial Port (USS) is used to configure (Parameterize) Masterdrives MCs either using OP1S or Drive Monitor in conjunction with Drive ES.

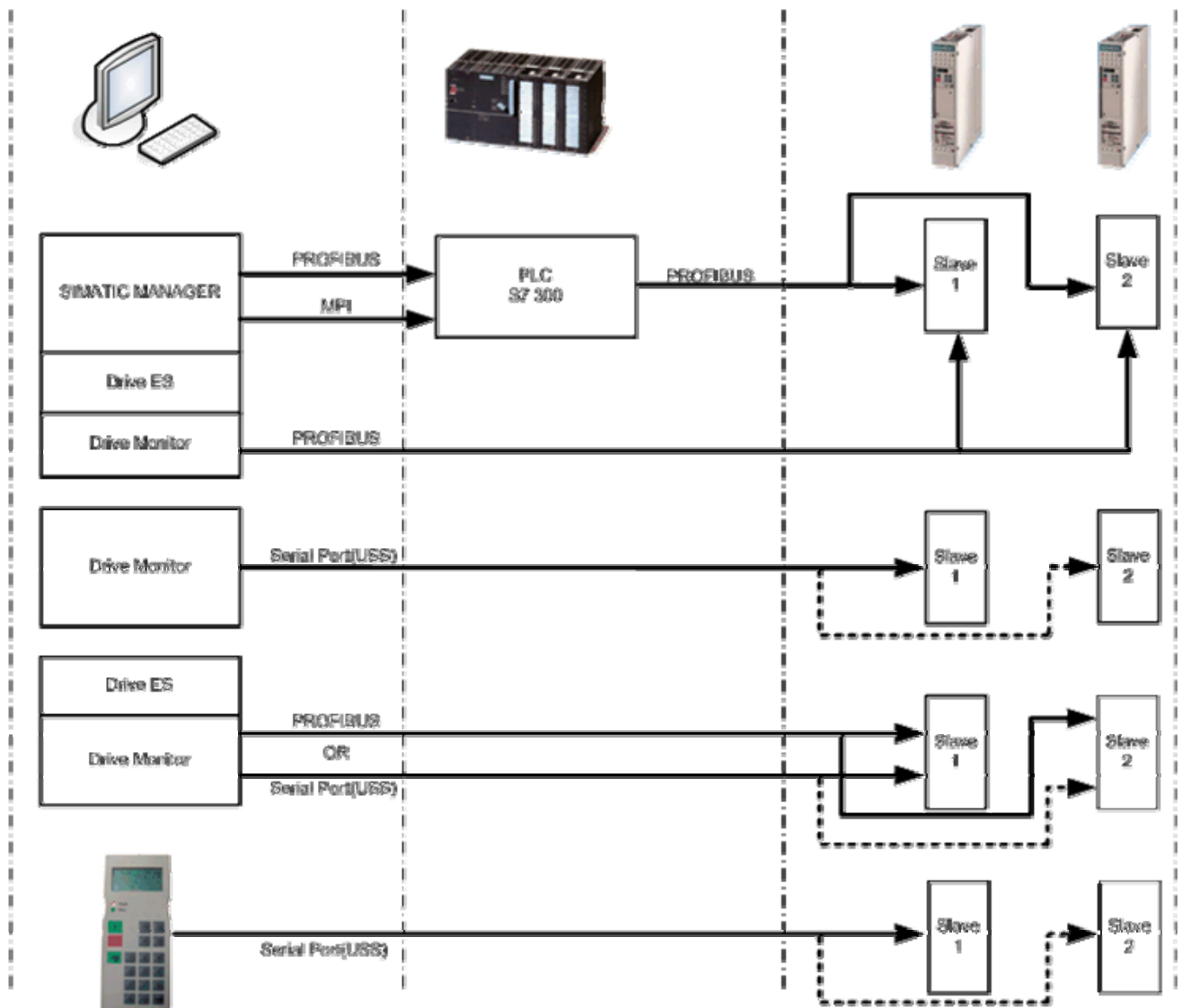


Figure 6.3 Means of Communications with Masterdrivers MCs

6.1 The Multipoint- Interface (MPI)

This bus system was mainly developed as a programming interface. MPI enables us to set communication with components that work for the ‘man/machine interface’ and for homogenous communication between automation devices. In this project MPI used to download the initial system configuration to the CPU.

The operation area from the MPI and PROFIBUS is divided into many areas, where MPI is considerably cost effective. This interface is already available in all SIMATIC S7 products as a standard bus system, therefore no “outside manufacturers” product is needed to be integrated for communication

- Up to 32 MPI nodes can be realized.
- Each CPU has a possibility of up to 8 dynamic communication connections for the basic communication to SIMATIC S7.
- Each CPU can operate up to 4 statistic communication connections for the additional communication to the PG/PC, SIMATIC HMI-Systems and SIMATIC S7.
- Data transmission speed 187,5 kbit/s or 12Mbit/s
- Flexible configuration possibilities in the bus or tree structure (with repeaters)
- Maximum wire length is 10km (with repeaters).
- Interface is RS485.

MPI can be used for simple linking in networks and enables the following forms of communication:

- **Global communications:** The networked CPUs can cyclically exchange data under one another.
- **Programming and diagnostic functions:** MPI executes these functions from other programmed devices/PCs to all networked PLCs. There the MPI interface of the CPU is directly connected with the internal communications bus (K-BUS) of the S7-300. The function modules (FM) and communications modules (CP) are switched directly over the MPI with the K-Bus connection from the PG.
- **Simple connection:** To connect operator panels/operator stations to the SIMATIC S7 PLCs.

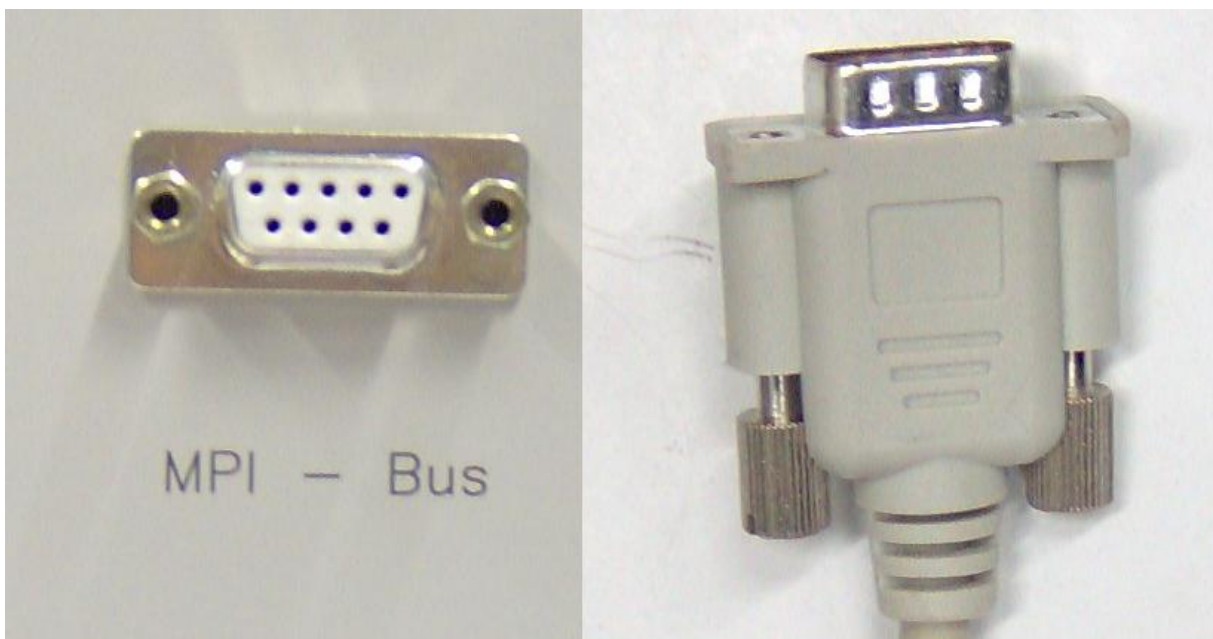


Figure 6.4 Connector for MPI bus

6.2 The PROFIBUS DP

The “**PROFIBUS**” (**PRO**cess **F**ield **BUS**) was developed in the beginning of 1991.

The goal of developing PROFIBUS was to develop a field bus system to link up a network of automation devices of the lowest field level from sensors and actors up to the process control in the cell level. It is optimized for SIMATIC communication, which has greater speed compared with other automation protocols used for data communication. Its availability for the bus systems of the management and cell level e.g.: industrial Ethernet and field level with PROFIBUS are some of the useful features. PROFIBUS DP uses Layer 1, and Layer 2, and the User Interface. Layers 3 to 7 are not developed, it is designed especially for communication between the programmable controller and the distributed I/O devices at the field level.

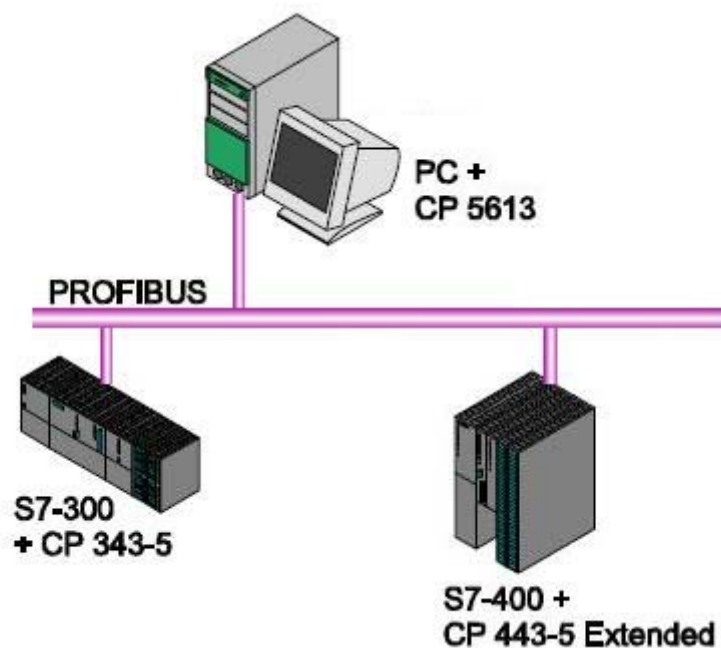


Figure 6.5 a sample view of PROFIBUS

PROFIBUS-DP specifications under the Norm 50170:

- The bus allocation occurs by the PROFIBUS-DP after the processing of ‘Token passing with supported master-slave’.
- Typical cycle time is between 5 -10 ms.
- Maximum number of nodes; 127
- Frame length of user data can be 0-246.

Transmission rate in Kbaud	9,6	19,2	93,75	187,5	500	1500	3000	6000	12000
Length per segment in m	1200	1200	1200	1000	400	200	100	100	100
Max. length in m	12000	12000	12000	10000	4000	2000	400	400	400
By number of bus segments:	10	10	10	10	10	10	4	4	4

Table 6.1 PROFIBUS transmission rates

- The bus configuration is modular expandable where as the peripherals and field devices are connected and unconnected during the operation.
- The data transmission occurs either over a 2 wire cable with a RS-485 interface or over a fibre-optic cable. We restrain ourselves here to the 2 wire cable data transmission possibility.
- The unprotected and twisted 2 wire cable (Twisted Pair) has a minimum cross section of 0.22 mm² and must be connected at the end with the shaft (termination) resistor.
- An area-wide network occurs by the PROFIBUS-DP through the compartmentalization of the bus system in the bus segments that can be connected over repeaters.

6.3 Ethernet

In SIMATIC NET communication system, Industrial Ethernet is the network for the management and cell level. Physically, it is an electrical network, which uses a shielded coaxial cable or twisted pair or fibre-optic cables with an optical network. The international standard IEEE 802.3 defines Industrial Ethernet. It is accessed using the CSMA/CD (Carrier Sense Multiple Access with Collision Detection) network access technique specified in IEEE 802.3

Ethernet has some important properties giving the following advantages:

- Fast start-up by most simple wiring methods
- High availability since existing installations can be expanded without any problems
- Practically unlimited communication performance since scalable performance through switching technology and high data rates are available if required
- Networking of the most varied fields of application such as offices and production facilities
- Company-wide communication through the coupling facility via WAN (Wide Area Network) as well as via ISDN or Internet
- No investment risk thanks to the continued and compatible further development
- Reserving of data at industrial wireless LAN (IWLAN). [1]

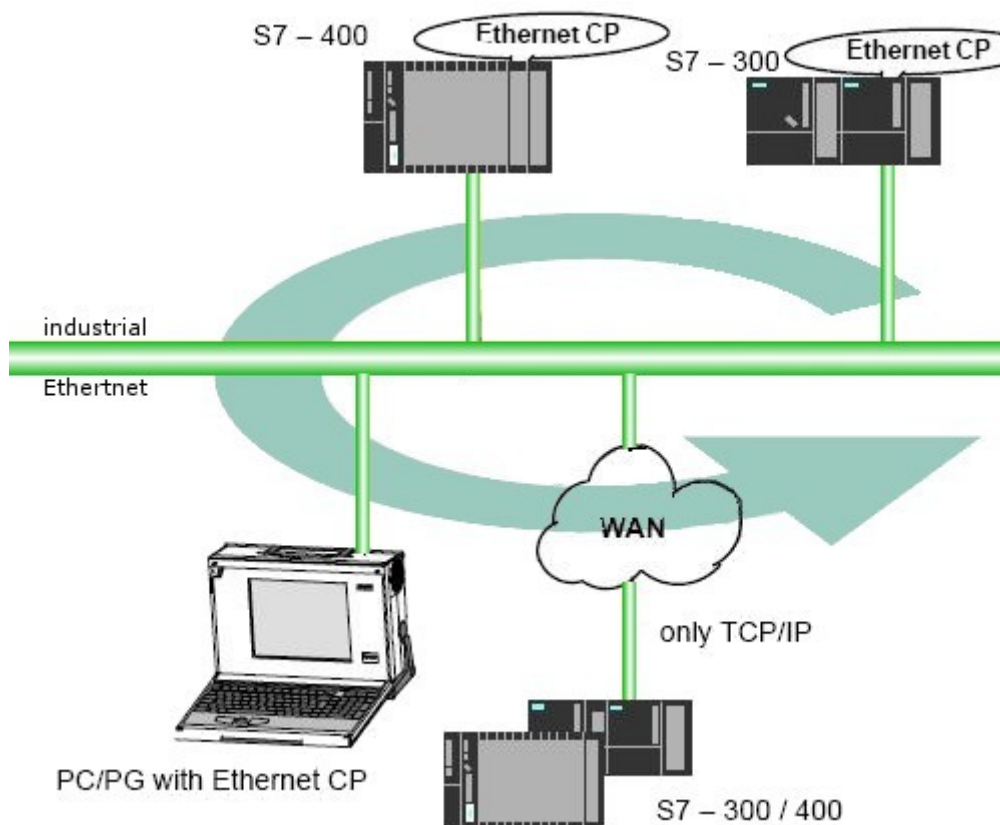


Figure 6.6 Industrial Ethernet

Types of Communications for S7 300/400 with an Ethernet CP	
Possible types of communication	Interfaces / Services / Protocols
<ul style="list-style-type: none"> • PG/OP communication • S7 communication 	with the protocols – ISO – TCP/IP (RFC 1006)
<ul style="list-style-type: none"> • S5-compatible communication 	with the SEND / RECEIVE interface and the protocols – ISO Transport – ISO-on-TCP (TCP/IP with RFC 1006) – TCP/IP – UDP – E-mail:
<ul style="list-style-type: none"> • PROFINet communication 	with the protocols – TCP/IP
<ul style="list-style-type: none"> • HTML process control with web browser 	with the protocols – HTTP / IP protocol
<ul style="list-style-type: none"> • File management and file access with FTP 	with the protocols – FTP / IP protocol

Table 6.2 Types of Communications for S7 300/400

7 MOTION CONTROLLERS

SIMOVERT MASTERDRIVES MC (Motion Control)

6SE7011-SEP50-Z and 6SE7012-OTP50-Z

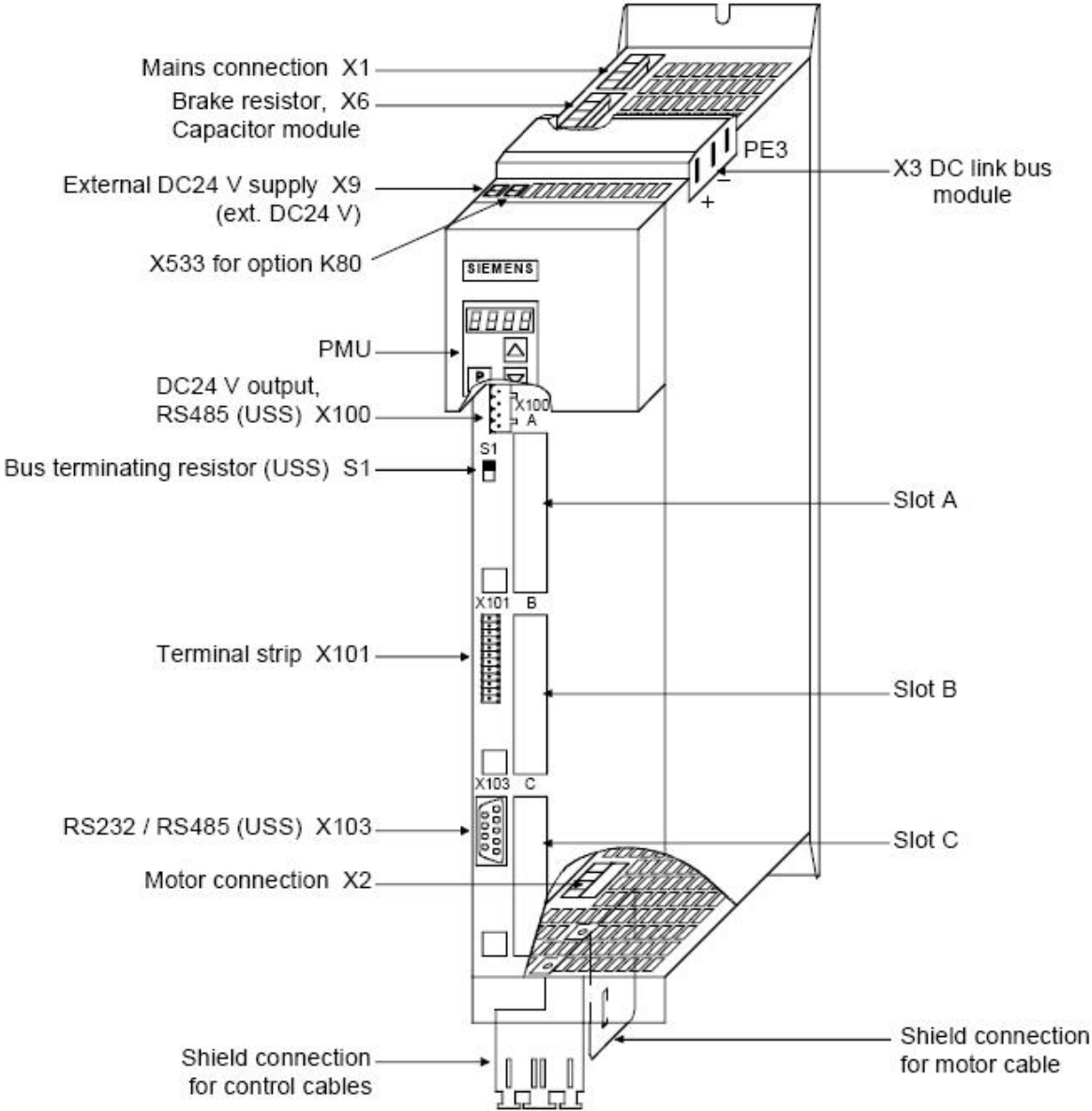


Figure 7.1 SIMOVERT MASTERDRIVES MC

Specifications	
Voltage range	3-ph. 380 - 460 VAC \pm 15%; 50/60 Hz \pm 6%
Power range	3/4 to 250 HP (0.5 - 200 kW)
Control Voltage	24 VDC
Overload	160% for 30 Sec. or 300% for 250 mSec. for Compact Plus
Ambient Temperature	32°F - 113°F (0°C - 45°C)
Installation Altitude	Up to 3300 feet (1000m)
Approvals	UL, CE, CSA, EN, IEC/VDE

Table 7.1 SIMOVERT MASTERDRIVES MC Specifications

The SIMOVERT MASTERDRIVES MC (Motion Control) belongs to the SIMOVERT MASTERDRIVES product group. This product group represents an overall modular, fully digital component system for solving all drive tasks posed by three-phase drive engineering. The availability of a high number of components and the provision of various control functionalities enable it to be adapted to the most diversified applications. [2]

The MASTERDRIVES Motion Control (MC) frequency converters are specially designed for industrial servo drive applications. In addition to the well-proven modular hardware concept, SIMOVERT MASTERDRIVES MC offers modular software featuring, freely interconnectable function blocks, and integrated technology functions to program specific applications (see Figure 2.8)

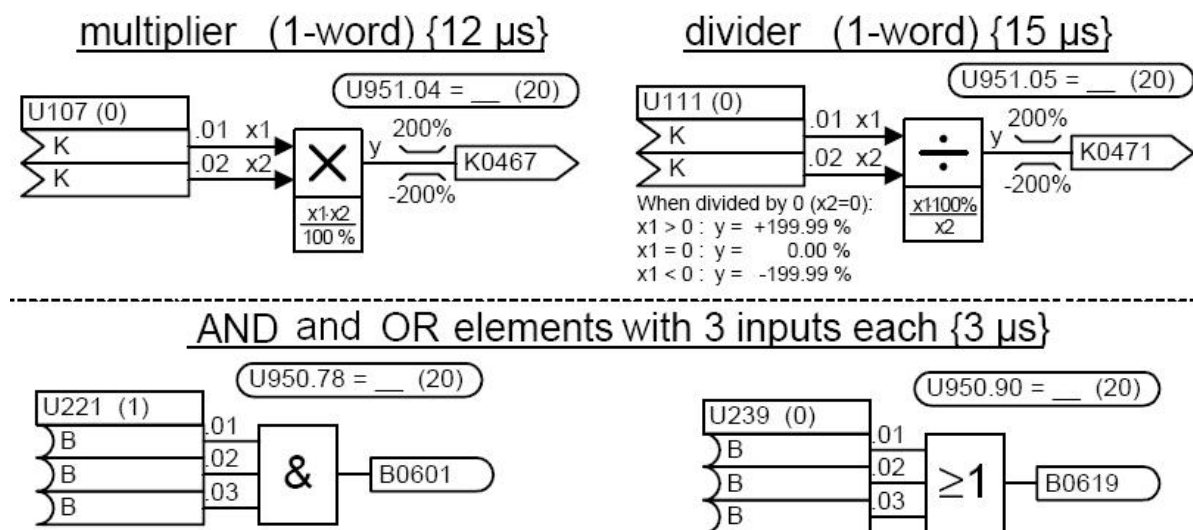


Figure 7.2 Free blocks [3]

MASTERDRIVES Motion Controllers compatible to;

- Communication, such as PROFIBUS
- technology
- operator control and visualization

Standard Features

One hardware platform to provide:

- Closed Loop Position Control
- Closed Loop Speed Control
- Torque control
- IGBT inverter bridge through entire power range.
- Automatic voltage adjustment within the power range.
- Packaging capability to be ordered for common DC bus configurations. Non-regenerative or full regenerative converter sections available as standard option.
- Ability to operate various speed/position feedback devices (Absolute Encoders, Optical encoders, Resolvers, and Pulse encoders).
- Able to operate various motors with one drive (Siemens synchronous servo, Siemens asynchronous servo, & standard inverter rated induction motor)
- 0.001 Hz set point resolution
- One analog input
- One analog output
- 4 programmable binary inputs or outputs (two can be used as high speed inputs with ~1 micro second sensing time)
- 2 dedicated binary control inputs (24 VDC Control)
- 2 separately addressable serial interface ports: 1 for RS485 and 1 for RS232/485
- Over 100 Warning and Fault messages for comprehensive protection.

7.1 Parameterization

In order to use SIMOVERT MASTERDRIVES one must program them first. There are large number of open-loop and closed-loop control functions, communication functions, as well as diagnostics and operator control functions implemented in the converters' and inverters' software interim of function blocks. These function blocks can be parameterized and freely interconnected depending on the applications. The interconnection is done through connectors

and binectors to exchange signals between individual function blocks. Connectors can be likened to storage locations which are used to archive "analog" signals. Connectors are designated with certain pattern, which includes connector name, the connector number and an identification letter (see Figure 7.4).

The identification letter depends on the numerical representation:

- K Connector with word length (16 bit)
- KK Connector with double-word length (32 bit, increased accuracy)

The binary (digital) output information is archived in **binary connectors** (binectors). Binectors can therefore be likened to storage locations used for storing binary signals. The designation of binectors is; the binector name, the binector number and an identification letter. The binector identification letter is B, and it only has two states "0" (logically no) and "1" (logically yes).

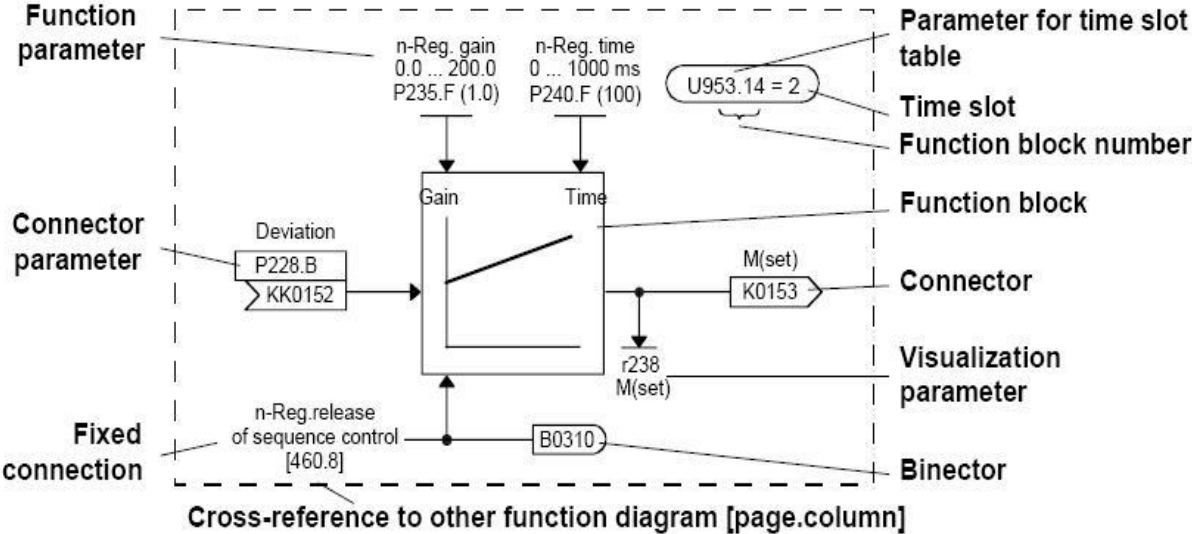


Figure 7.3 Function Blocks

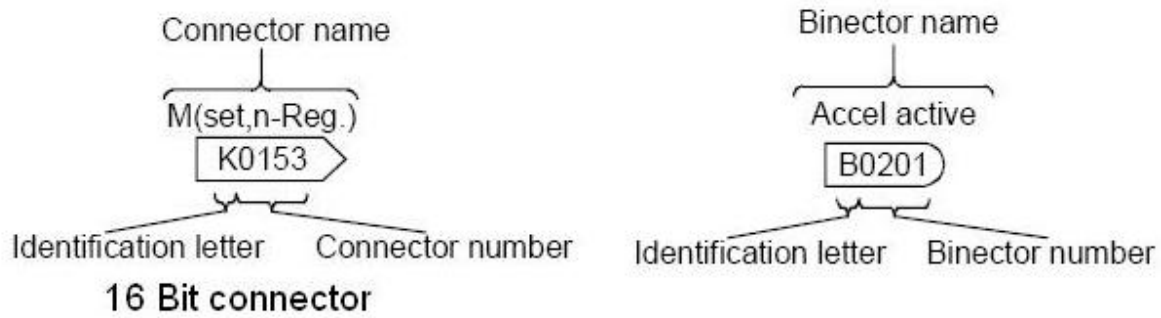


Figure 7.4 Connectors and binectors

To connect two function blocs is as simple as assigning a connector to a parameter. Ass it is seen from Figure 7.5 function block A is connected to function block B by assigning connector KK0152 to parameter P228.01.

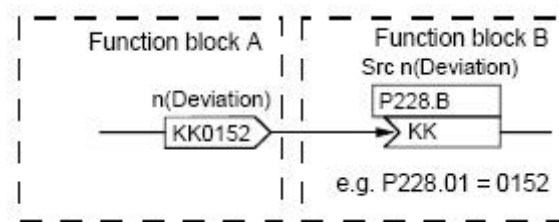


Figure 7.5 Connecting two function blocks

Parameterization can be done in 4 different ways;

1. **PMU**
2. **OP1S**
3. **Drive Monitor (Stand alone through MPI connection)**
4. **Drive Monitor (With S7-Project through PROFIBUS)**

Even though it is more comfortable to use OP1S compare to PMU, there are limitations to the usage. Here are the list of menus that can be used via OP1S or/and PMU (Figure 7.6)

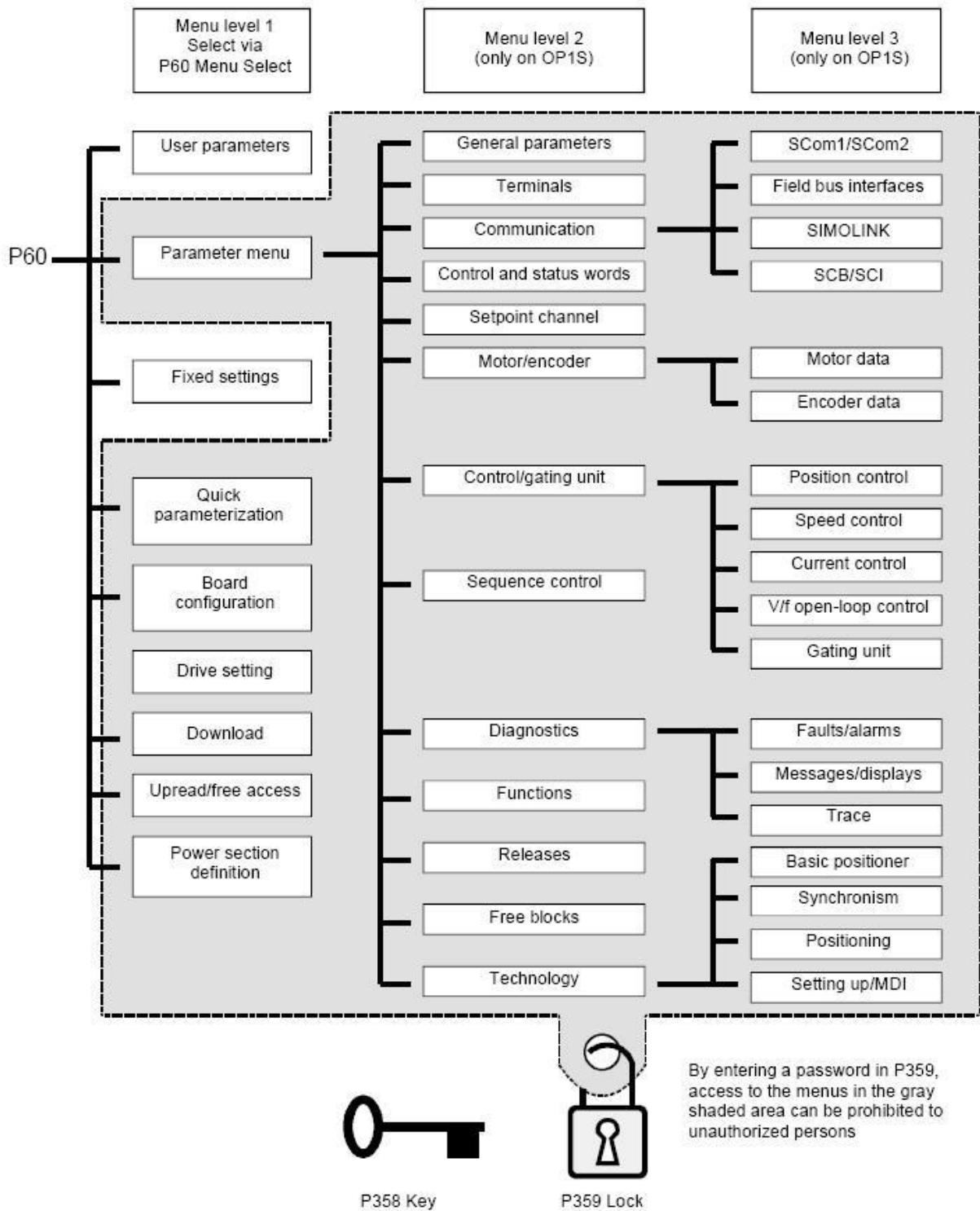


Figure 7.6 Parameter menus [4]

7.1.1 Parameterization via PMU

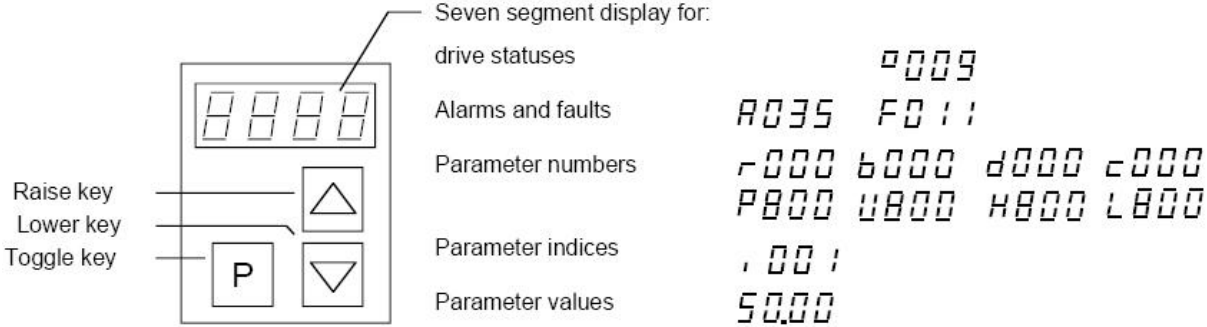


Figure 7.7 PMU

As it is seen from the Figure 7.7; PMU has a seven segment display, a toggle key, and up and down buttons. By default the drive status would be displayed, so in order to enter or modify a parameter one should follow below steps;

- Press the toggle key first to display the parameter number.
- Pres raise key to change the parameter number.
- If the parameter chosen then pres toggle key choose parameter index
- Pres raise key to select right index
- Pres toggle key to select parameter value.
- Press the toggle key to exit

7.1.2 Parameterization via OP1S (6SE7090-0XX84-2FK0)

The operator control panel (OP1S) is an optional input/output device which can be used for parameterizing and starting up the units. OP1S is more user friendly than the PMU; instead of raising or lowering parameters by PMU it is possible to jus type the number and modify it. Besides that it has some very useful features that make OP1S better choice, for example it has a non-volatile memory and can permanently store complete sets of parameters, which enables us to back up parameter sets. Therefore parameterizing same devices on the network becomes very fast and easy.

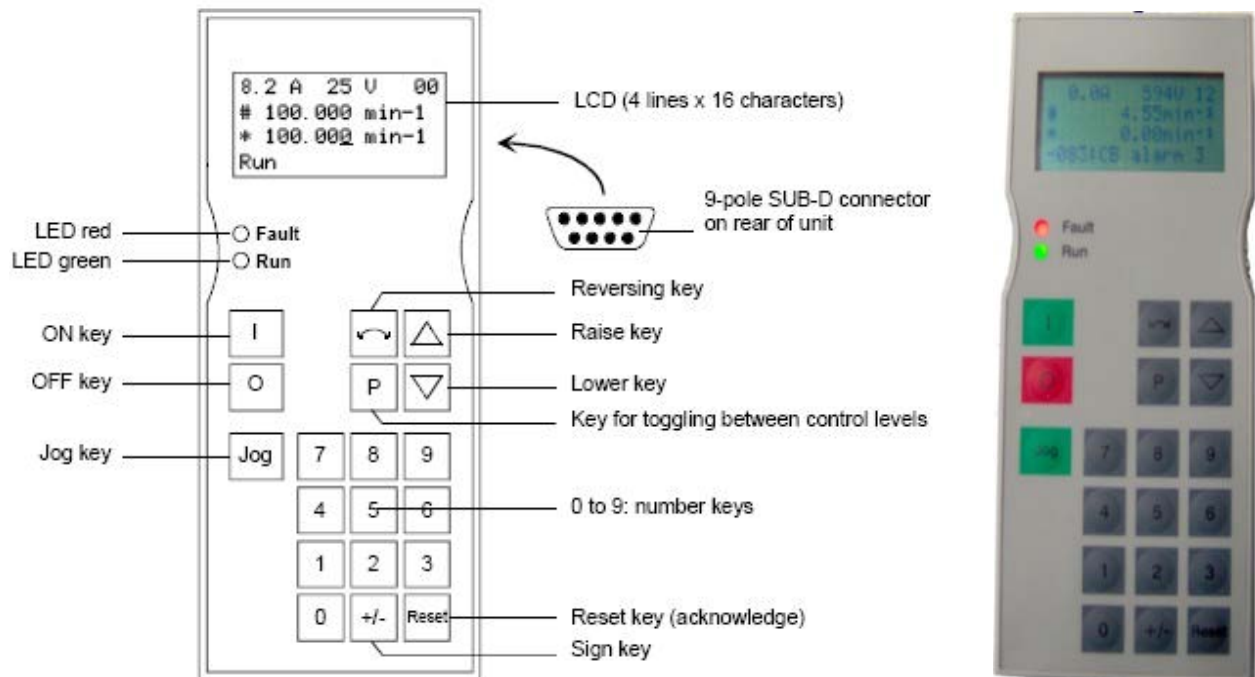


Figure 7.8 OP1S

The OP1S communicates to drivers via a serial interface (RS485) using the USS protocol. It acts as master and the drive as slave during the communication. The operation rate is between 9.6 kBd and 19.2 kBd. There are 32 nodes on the network (addresses 0 to 31) for OP1S to communicate. During the start up it detects all the slaves. Through menu one can jump between slaves. The plain-text display can be operated in German, English, Spanish, French, and Italian.

The basic menu is the same for all OP1S units. By pressing the “P” key the following selections can be made:

- Menu selection
- OP: Upread
- OP: Download
- Delete data
- Change slave
- Config. slave

Due to the screen size not all the lines can be shown at the same time. It is possible to scroll the display as required with the "Lower" and "Raise keys.

```

0.0 A 0 V 00
# 0.00 min-1
* 0.00 min-1
Ready
    
```

This is the default view of OP1S screen, which displays operation information. In order to go to menu select the "P" key must be pressed once.

```

MotionControl
*Menu selection
OP: Upread
OP: Download
    
```

Now, by pressing the up and down keys the desired menu can be highlighted. To choose the menu that you want, and then the "P" key has to be pressed again.

```

      ↗ P
MotionControl  Change slave
  Download      Address:00
  Delete data
#Change slave
    
```

The cursor is under the slave number (00). Just type the new address and then pres "P" key.

Figure 7.9 OP1S Screen shots

I you want to go back to previous menu just pres "Reset" key once. For more information on OP1S module, please check the Compendium (e.g. mc166_kompend_e.pdf).

7.1.3 Parameterization via Drive Monitor (Stand alone)

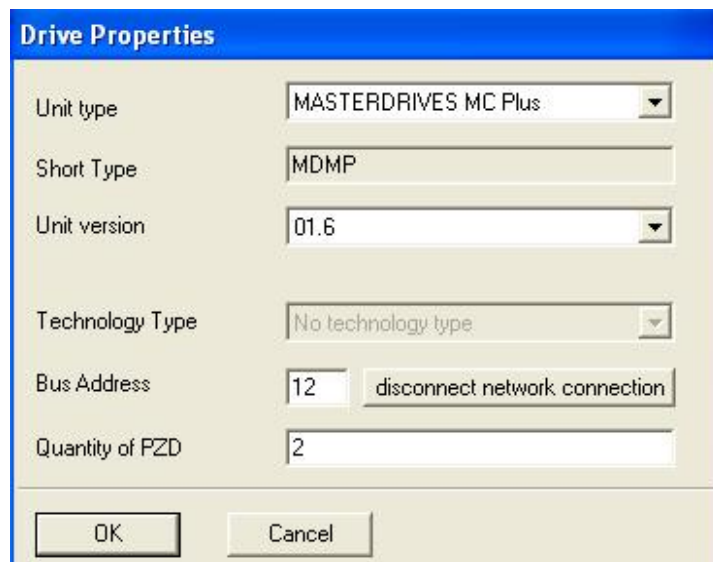
Here I would like to describe how to start and configure the drive monitor to communicate with the DP slave, and then upload parameter sets from the DP slave for back up reasons. At the beginning we start with a blank page, and import the parameter set from the back up. Also some of the parameters through drive monitor and some through PMU must be configured. With Drive Monitor parameters like drive address, display language, and etc can be changed. Parameters like menu select are configured by PMU to have access to the drivers.

7.1.3.1 Install Drive Monitor



DriveMonitor

- Double click **DriveMonitor** icon to start drive monitor.
- Press the new empty page button to have a parameter set, which will be blank page, because the unit is not connected yet.
- You will see the below window to set bus address, and unit version (the firmware version of the Masterdrive MC, which is 1.66). When you press okay button; you will be asked to save the parameter set. Choose the location, and a name, and then save it.

A screenshot of the 'Drive Properties' dialog box. The dialog has a blue title bar and a light beige background. It contains several fields: 'Unit type' is a dropdown menu with 'MASTERDRIVES MC Plus' selected; 'Short Type' is a text box with 'MDMP'; 'Unit version' is a dropdown menu with '01.6' selected; 'Technology Type' is a dropdown menu with 'No technology type' selected; 'Bus Address' is a text box with '12' and a button labeled 'disconnect network connection' to its right; 'Quantity of PZD' is a text box with '2'. At the bottom are 'OK' and 'Cancel' buttons.

Field	Value
Unit type	MASTERDRIVES MC Plus
Short Type	MDMP
Unit version	01.6
Technology Type	No technology type
Bus Address	12
Quantity of PZD	2

Figure 7.10 Derive Properties

- Now you will be prompted with the below window. Choose communication → Serial Interface ½ from the list on left hand side.

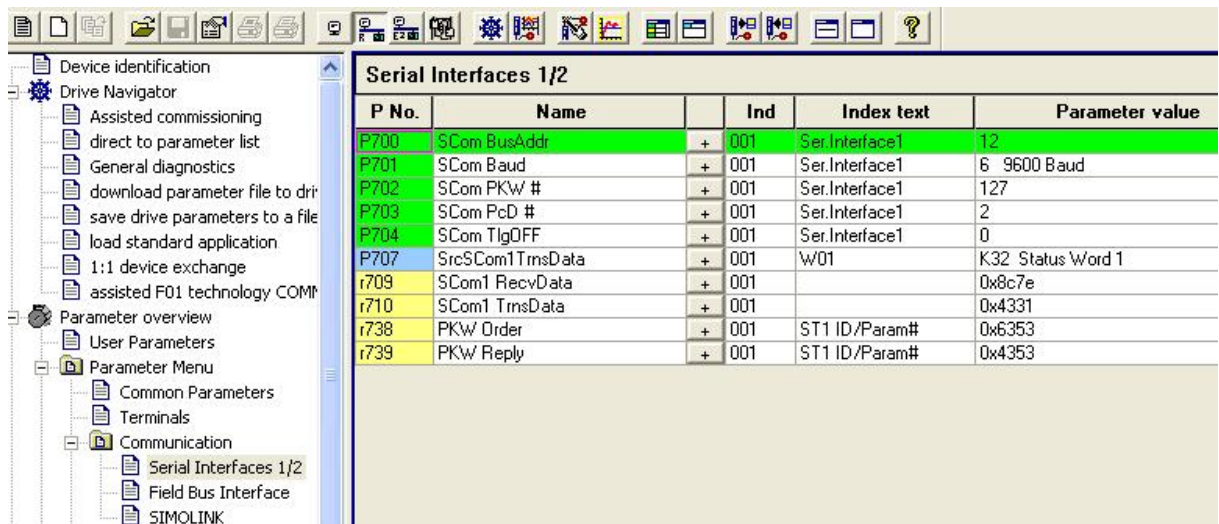


Figure 7.11 Drive Monitor Communication

- Double click on P700 parameter, and as it is seen in the below figure set the serial port buss address. In this case it is set to 12.

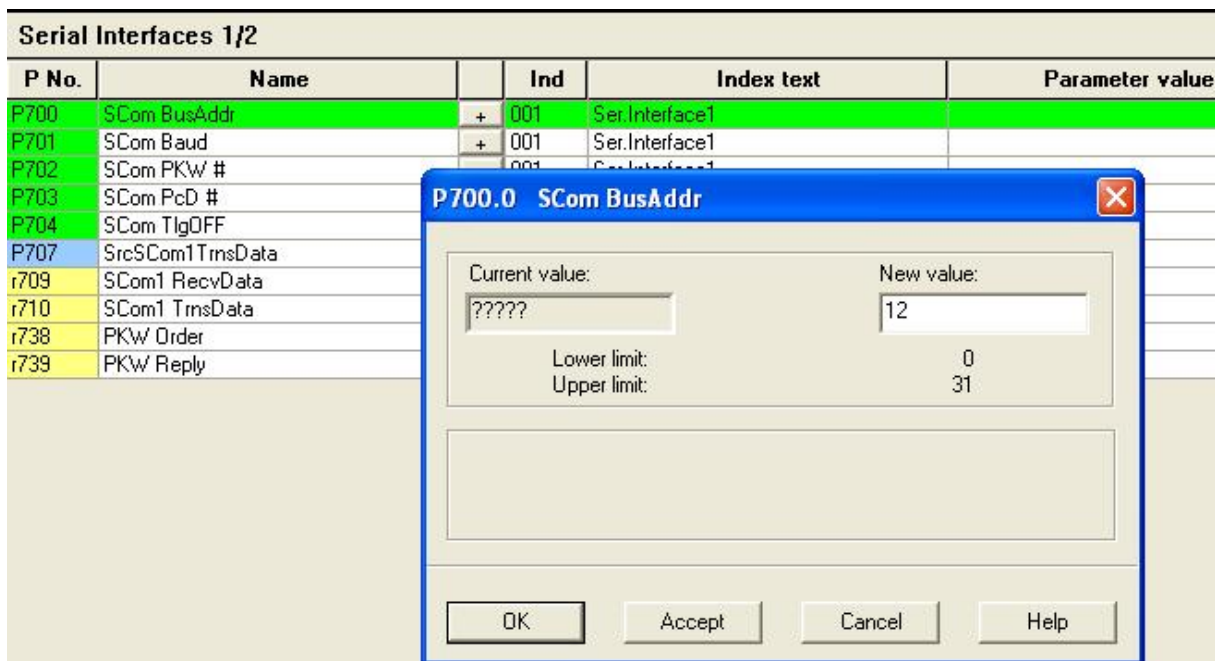


Figure 7.12 Set serial port bus address

7.1.3.2 Parameterize the slave (Masterdrives MC)

In order to be able to connect drive monitor there are few parameters have to be configured by using PMU.

- Set parameter P053 to 7. Parameter P053 is displayed in binary format. Each bit has a certain meaning, as displayed blow picture.

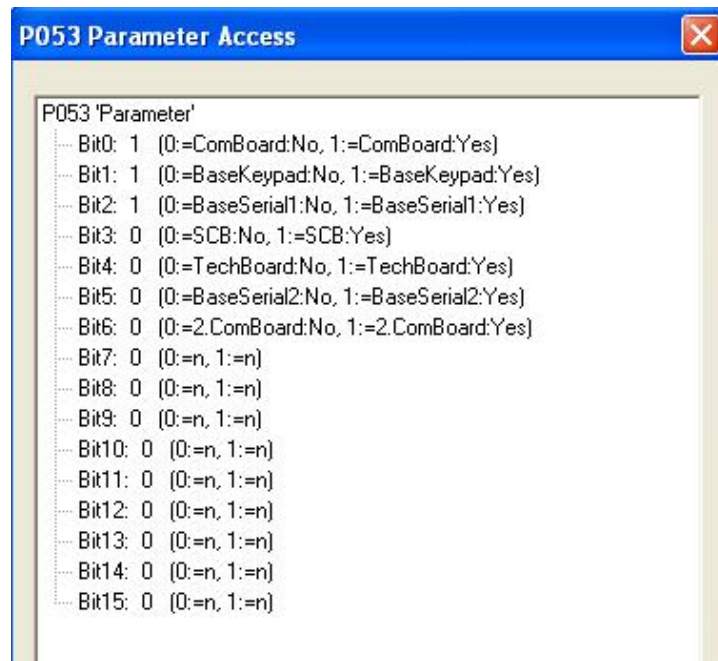


Figure 7.13 Parameter access

- Set parameter P060 to 7, which gives you the right for Upread/Free access.
- Parameter P700, which enables us to define the serial port buss address. Set this parameter to the values you used to set serial port bus address previously in Drive monitor.
- now press Online (Write RAM) button on drive monitor menu bar to go online, which means to connect to the DP slave



Figure 7.14 Drive Monitor online button

- If you are having problems with connecting to the DP slave with an error message wrong COM port. Due to security reasons COM ports are not open to communication in

Windows XP. Which means your operating systems' settings has to be changed. Make sure you are logged in as administrator.

7.1.4 Parameterization via Drive Monitor (SIMATIC Manager)

In this case the PROFIBUS is going to be used for connection, and then the Drive monitor is going to be opened through S7-Project. As it is seen in Figure 7.15 on the top; there are two ways to connect to the CPU; PROFIBUS or MPI, and one way to the DP slaves, PROFIBUS. Please see the Appendix 1 for hardware configuration and building as S7 project. In order to open Drive Monitor:

- start SIMATIC Manager
- Open the project that previously configured. If there is no project then follow steps in HW configuration to configure hardware in a S7-Project, including adding the slaves that needs to be connected.
- In S7-Project open parameter set folder by opening MASTERDRIVES CBP → MASTERDRIVES MC Plus→Parameter. Of course the folder is empty, because there are no parameter sets have been saved.

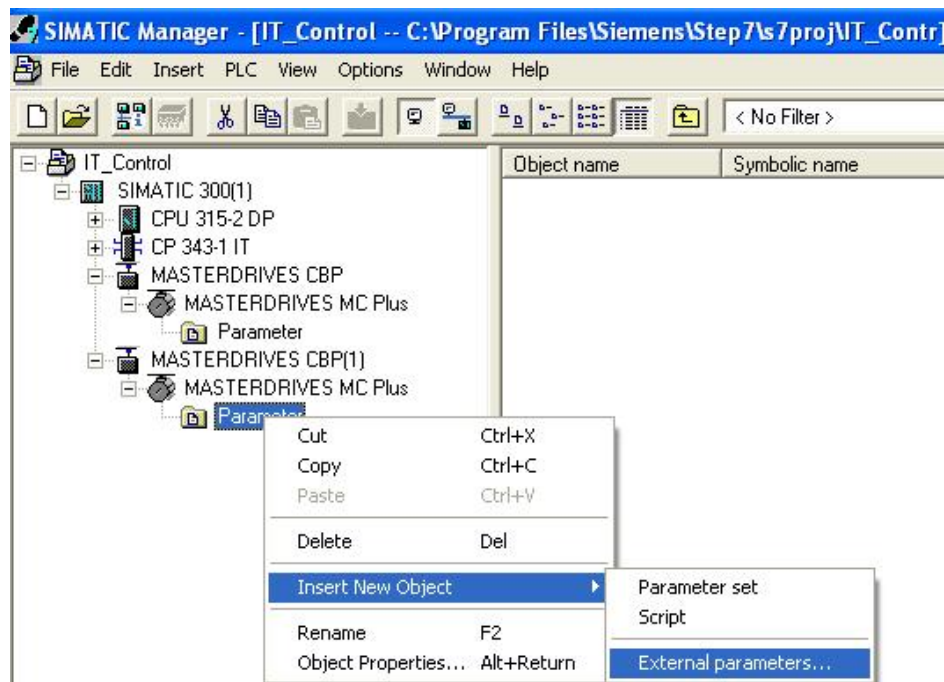



Figure 7.15 Project insert parameter set

- Right click on the parameter folder → Insert New Object → Parameter set or External parameters (see Figure 7.15). If it is the first time that device been connected, there are no parameter sets have been saved, and then chose “Parameter set” menu.
- After “Parameter set” is been chosen, now by double clicking on it one can open the parameter set with Drive Monitor.
- Select the Communication folder in parameter view in Drive Monitor then click in Field Bus Interface
- To set PROFIBUS address use parameter P918, and use bus address between 0 and 127. This bus address should be same as the one chosen during the hardware configuration.
- Use PMU to change buss address on drive by setting parameter P918’s value to the value in pervious step.
- Save your project and go on line. Now you will see the Drive monitor below figure, if not the pres the  button in drive monitor, or in the menu choose → Drive Navigator → Drive Navigator.

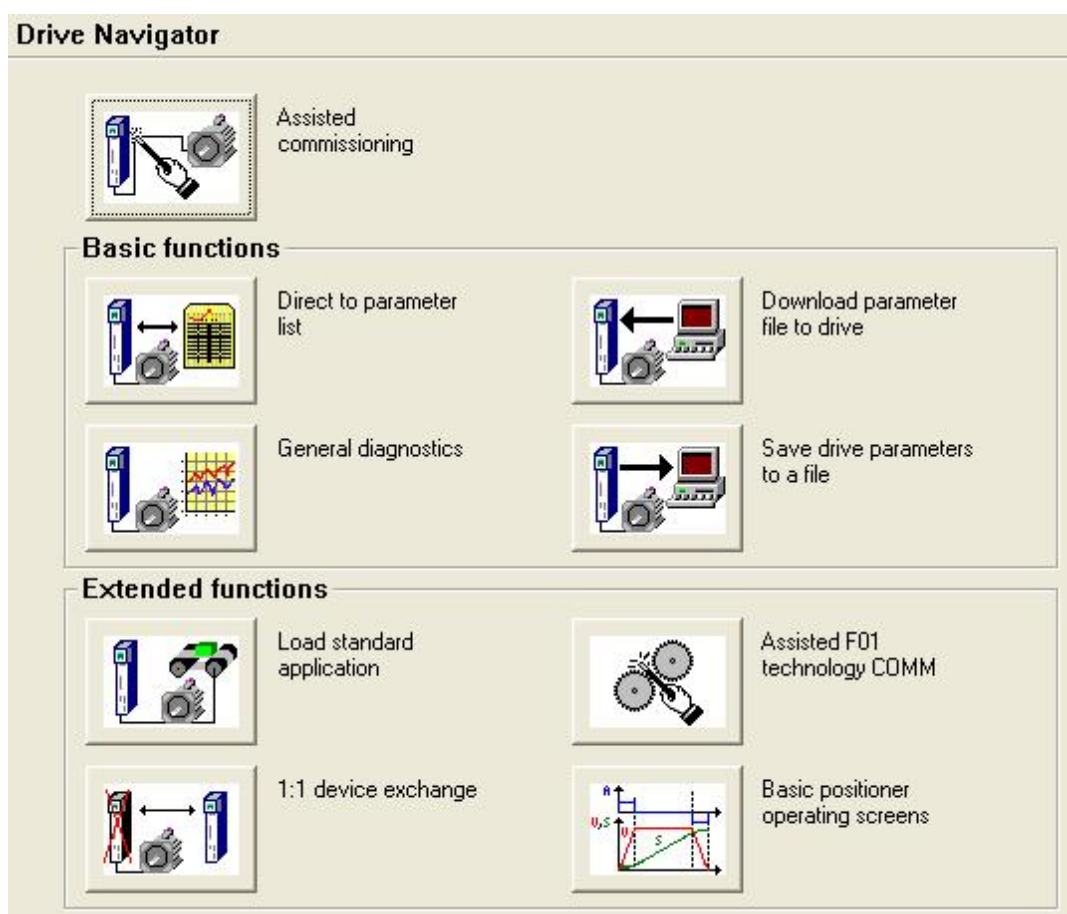


Figure 7.16 Drive Monitor Drive Navigator

- From this view by pressing “Save drive parameters” button the parameter in the drive can be saved to PC.

7.2 Parameterization Steps

Until now parameterization methods were discussed. After choosing one of the above methods one can start parameterize the DP slaves. The most convenient method is to use Drive Monitor, because it enables to see parameters as a complete list or in different categories and subcategories e.g.: Communication, Motor/encoder Data. For example Communication parameters are listed under three different categories;

1. Serial Interface 1/2
2. Field Bus Interface
3. SIMOLINK

In every category only the relevant parameters are listed, it therefore makes our job easier.

The method 4 (“Parameterization via Drive Monitor with SIMATIC Manager”) is chosen to explain to parameterize DP slaves. Before starting to parameterize the DP slaves one should change few parameters with PMU to make thins simpler (underlined parameter are chosen in this project’s configuration);

P050 Parameter to define the language

Function parameter for setting the language in which texts are to be displayed on the optional OP1S user-friendly operator control panel, on Drive Monitor.

P050 = 0 = German
 1 = English
 2 = Spanish
 3 = French
 4 = Italian

This parameter is not reset during factory setting!

P060 Parameter to choose the user parameter menu please see Figure 2.12

P060 = 0 = User parameter (selection of the visible parameters in [P360](#))
 1 = Parameter menu
 2 = Fixed settings (for factory settings)
 3 = Quick parameterization (changes to "Drive Setting" state)
 4 = Board configuration (changes to "Board Configuration" state)

5 = Drive setting (changes to "Drive Setting" state)
6 = Download (changes to "Download" state)
7 = Upread/Free access
8 = Power section definition (changes to "Power section definition" state)

P053 Function parameter for releasing interfaces for parameterization.
P053 = 0 Hex = None
1 Hex = Cbx communication board
2 Hex = PMU operator control panel
4 Hex = Serial interface (SCom/SCom1), also OP1S and PC
8 Hex = SCB serial input/output modules
10 Hex = Txxx technology board
20 Hex = Serial interface 2 (SCom2)
40 Hex = Second CB board

0000000000111 (Binary) = 7 (Hex) is chosen.

P918 To define PROFIBUS bus address. 12 is chosen in this project.

P700 To define Serial port bus address. This is optional in this stage, but one can set it in order to use OP1S as well. I have chosen 12 for this project.

As it is seen in the parameter P060 = 7, Upread/Free access is enabled. For other tasks one must choose the relevant menu option.

PS: At first the Masterdrive MC master has been configured, and the above buss addresses are only used for this drive. For the second DP slave different bus addresses must be used.

7.2.1 Main Steps for parameterization

The Figure 7.17 shows the steps of parameterization;

1. Power section definition (P060 = 8)
2. Board definition (P060 = 4)
3. Drive definition (P060 = 5)
4. Function adjustment.

7.2.1.1 Power section definition (P060 = 8)

The power section definition is necessary for compact, chassis and cabinet units. It informs the control electronics about the power section that working with. Power section definition is ready as delivered state, and it is therefore only necessary when the firmware is replaced with a new parameter set.

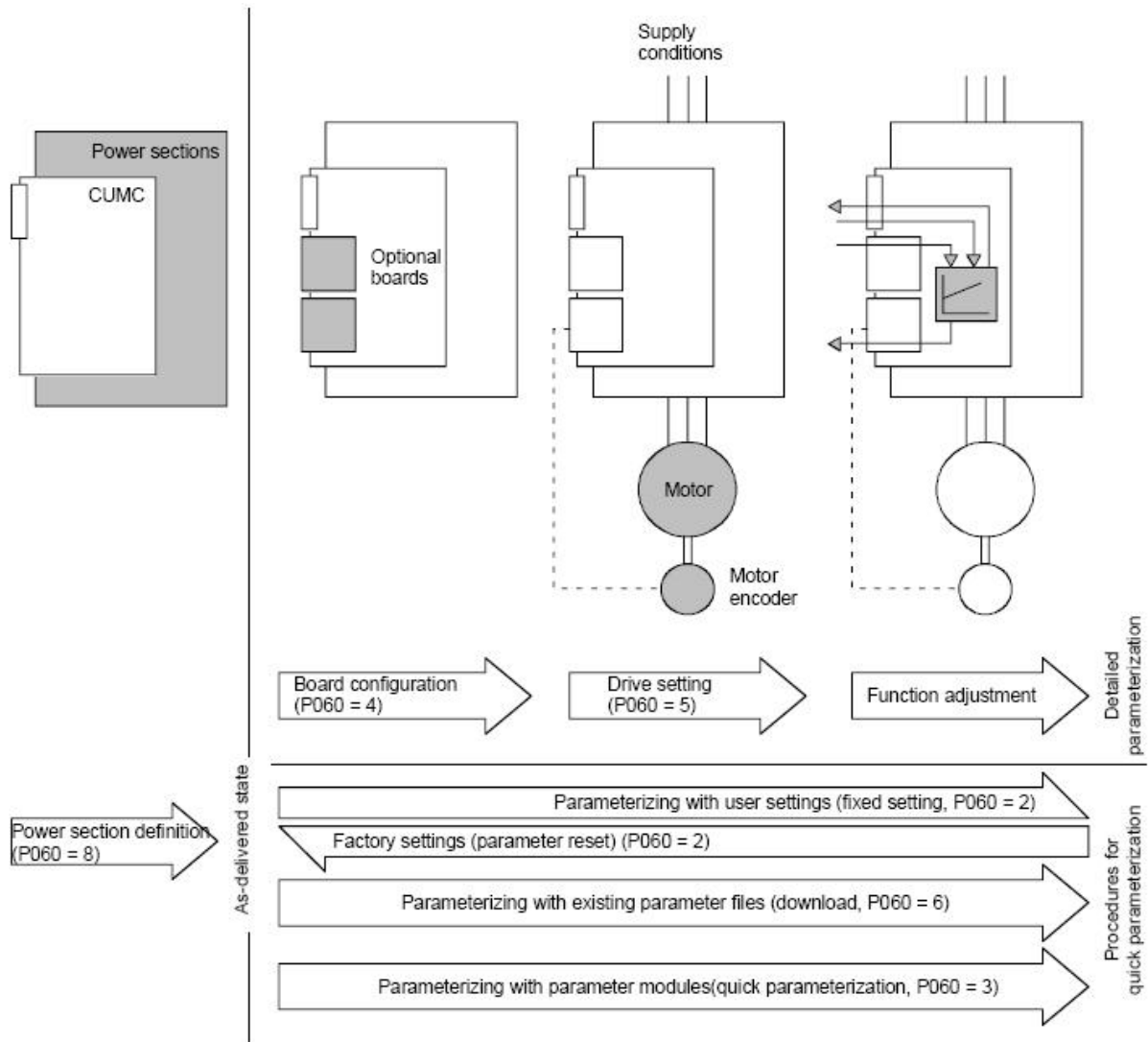


Figure 7.17 Detailed parameterization steps

P060 = 8 Power section definition

P070 = the relevant PWE is to be chosen depending on the unit that is going to be parameterized. This can be defined by looking at the order number of the unit then check in the list provided by siemens. For example the units that are been used in this project are;

	Order Number	In [A]	PWE
Frequency Converter,	6SE7011-5EP50	1.5	1

Compact PLUS AC-AC type			
Inverter, Compact PLUS DC-AC type	6SE7012-0TP50	2.0	2

Table 7.2 Power section

Choose

P060 = 1 Return to parameter menu

7.2.1.2 Board Configuration (P060 = 4)

The visualization parameter r826.x is used for displaying the board codes. Here are board codes that for the Inverter Compact Plus DC-AC;

r826	PCB Code	.	001	Basic board	94
			002	Slot A	143
			003	Slot B	161
			004	Slot C	115

Table 7.3 Board configuration

Looking at the tables provided in Compendium V1.66, and board codes listed in r826.x we can find out the type of board used in the Inverter.

Board	Slot	Parameter Value (PWE)	Significance
CUMC		94	Basic board
CBP	Slot A	143	Communication board PROFIBUS
SLB	Slot B	161	SIMOLINK bus interface
SBR 2	Slot C	115	Sensor board resolver 2

Table 7.4 Boards to be configured

Of course when parameterizing for first time there won't be any codes listed in parameter r826.x. One must know which boards are installed in the device, and then find their corresponding parameter values in Compendium. Below figure is showing the steps how to configure CBx (CBP) board, which is going to be the PROFIBUS bus address, and going to be used for communicating with the drive.

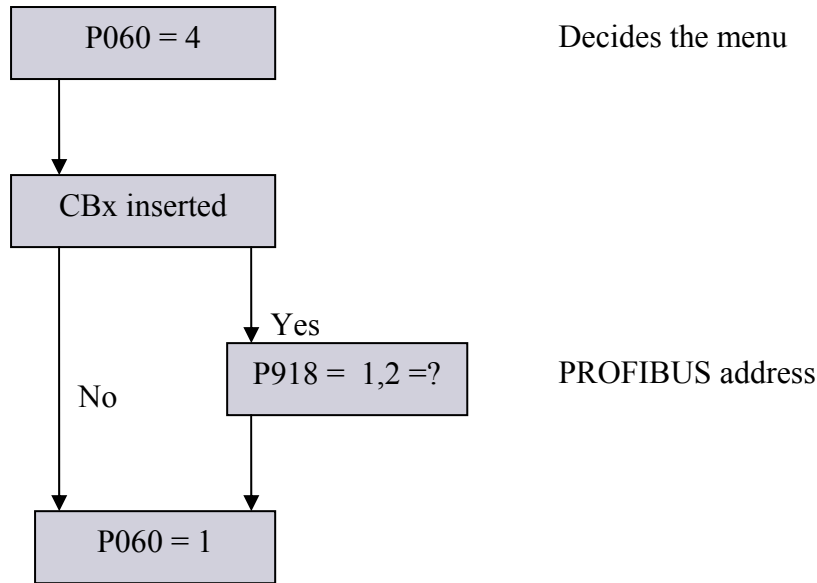


Figure 7.8 CBP board parameterization

7.2.1.3 Drive Setting (P060 = 5)

The control electronics will be configured depending on the motor and motor encoder. In this section motor data (normalization values) would be entered (parameterized) such as;

- motor type
- motor voltage
- motor current
- motor speed
- stator resistant
- motor torque

In addition, the type of motor control (V/f open-loop control or vector control) and the pulse frequency are selected. In order to do drive setting; one must carefully study the motor parameters and the control theory, because the performance of the application will greatly be affected from it.

- P060 = 5 Select “Drive Setting” menu
- P071 = ? Input unit line voltage in V (540 V)
- P095 = ? Input motor type
 - 0 = No motor connected
 - 1 = Synchronous servomotor 1FK6/1FT6
 - 2 = Induction servomotor 1PH7/1PL6/1PH4

3 = Synchronous servomotor general

4 = Induction motor general

If Siemens servomotors are used and 1 or 2 is entered, the connected motor can be directly selected in P096 and P097. The stored motor data are then taken automatically from an internal list. If other motors are used (entry of 3 or 4), the motor data must be entered separately. [4]

P096 = ? Enter Function parameter for selecting a 1FK6/1FT6 synchronous servomotor from the internal list of motors. For parameter values, see annex "Compendium".

Input P096	Motor order number (MPRD)	Speed n_n [rpm]	Torque M_n [Nm]	Current I_n [A]	Number of pole pairs
1	1FK6032-6AK7	6000	0.8	1.5	3

P128 = 3 A Max current

P130 = SBR Res. 2-Pole Select the motor encoder

P290 = 0 Current control Select the type of current control

P325 = 2 Input voltage boost

.
. .
.

For more detailed information see the compendium.

7.2.1.4 Function adjustment

Up to now in parameterization hardware characterization has been done. Depending on the application one must select, and interconnect suitable function blocs in the unit. The function blocs are provided for study at the end of the compendium. As indicated earlier in this chapter through the connectors and the binectors certain function blocks can be connected to build a specific application.

8 Modern Control

8.1 Synchronous Operation

Synchronous operation is a control strategy that is widely applied to industrial field for high efficiency and precision. For example in fully automated print houses motors highly synchronized. Synchronization does not mean operation under the same speed. Depending on the gear diameter of each motor, speed shall be adjusted so the belt feeder moves smoothly.

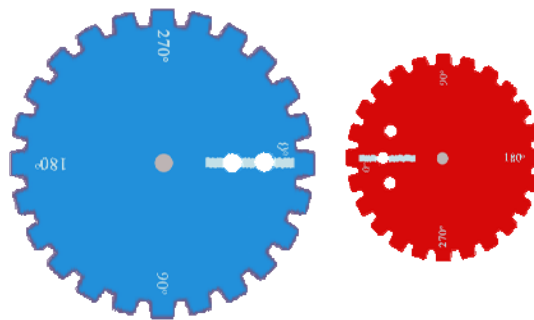


Figure 8.1 Gears

In the training setup there are LEDs synchronously flashing behind the gears. Every time the hole is in front of the LED, it flashes so that one can visually see that it is synchronized. Also there are LEDs flashing in between the gear teeth, which are confirming the synchronization of the two gears.

There are two different methods for doing the Positioning;

1. Basic positioning
2. Technology Option F01

8.1.1 Basic Positioning

The basic positioning can be done through the “Basic Positioner”, which is implemented by connecting the free blocs. It provides operations modes to move axis from point-1 to point-2.

Advantages of basic positioner:

- Cost neutral (with basic unit functionality)
- Easy to understand (basic commissioning)
- Continuous setpoint evaluation (during constant transfer)
- Control/checkback interface using BICO technology (e.g. PLC connection)
- Mode change on the fly (REF, POS, SETUP)

- Lower calculation time loading
- Lower project engineering costs
- Greater freedom for applications
- SIMATIC S7 not absolutely necessary [3]

The basic positioner made of the three free blocks, and it is available at no extra cost with the same functionality as the basic unit to provide a solution for “basic” positioning applications.

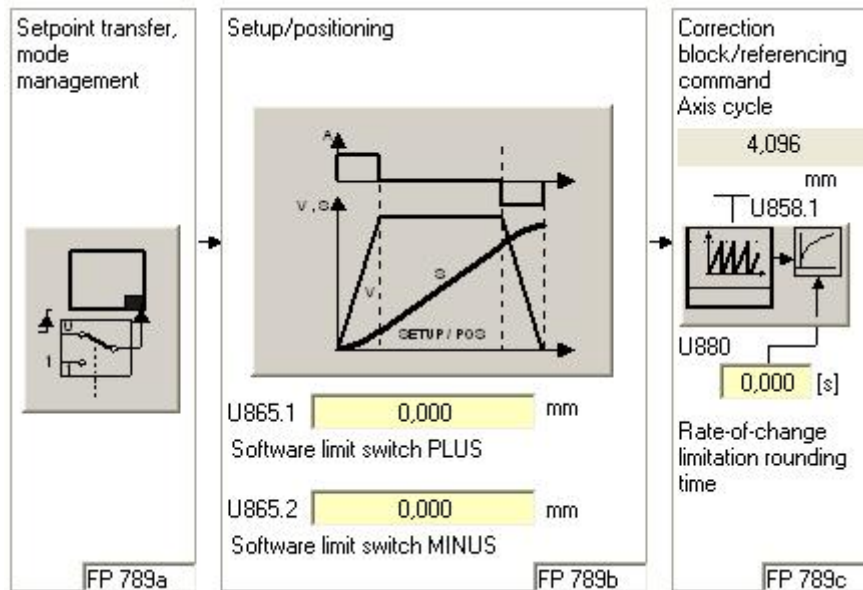


Figure 8.2 Three free blocks

Setpoint transfer and mode management [FD789a]

Setpoint transfer block with mode management and edge-controlled setpoint transfer for consistent data transfer.

Setup/positioning [FD789b]

Setup/positioning block that traverses a specified path relatively or absolutely using the specified deceleration, acceleration and speed.

Correction value/homing [FD789c]

Correction block, which provides the position correction and position setpoints for linking to the position controller and the position detection (see overview diagram FD788, FD788a for linking to the basic unit). [4]

8.1.2 Technology Option F01

Technology option F01 is another method to do positioning, which can only be used on a MASTERDRIVES unit if the unit was supplied with the option by the manufacturer. Even though it is supplied with option that does not mean it is ready for use. For that it should be enabled, which involves in extra costs. By checking the display parameter n978 one can see if the F01 option is present;

Free Parameterization		System/ForPrint				
P No.	Name		Ind	Index text	Parameter value	Dim
n978	Tech Release				1	

- n978 = 0 ==> F01 has been disabled
- n978 = 1 ==> F01 technology option has been enabled
- n978 = 2 ==> F01 technology option has been enabled for 500 hours

Function Diagram Sheet 850 shows that, how one can enable the technology option on a permanent basis or for a 500-hours trial period.

The Technology Option F01 has the positioning and synchronizations functions. Positioning can be done with or without help of the Technology Option F01.

The "Technology Software F01" software option contains the following functions:

- Positioning
- Angular synchronization

A MASTERDRIVES MC Power converter with the "technology" software option can be ordered by specifying the MLFB extension "F01".

8.1.3 General functions of Technology Option

Here are general functions that are included to the Motion Control technology software:

- **Linear axis:** With fixed stops and a maximum traversing range of 1000 m with a resolution of 1 μ m. Software limit switches are evaluated. Here is a transfer carriage, which makes a good example of a linear axis.

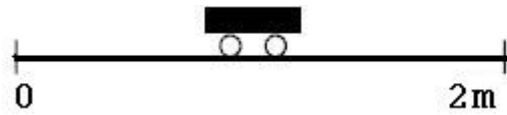


Figure 8.3 Linear axis [3]

- **Rotary axis:** Rotating infinitely, without fixed stops, with specification of the direction or direction of the "shortest path".



Figure 8.4 Rotary axis

- **Roll feed:** (infinitely rotating rotary axis with "cut-to-length" function). The figure shows the roll feed as used in a cutting device[4]:

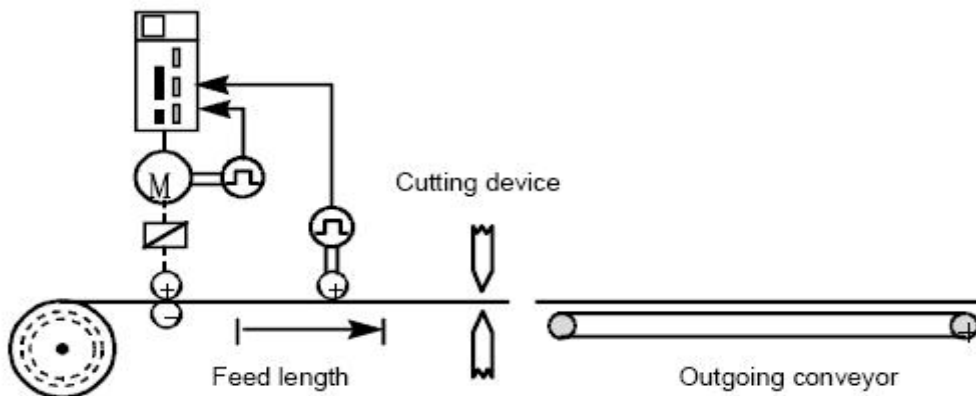


Figure 8.5 Roll feed [3]

8.1.4 Positioning

The MASTERDRIVES MC servo converter has an integrated positioning control system, which includes the following functions (for the detailed explanation of the functions please see the compendium):

<u>Function name</u>	<u>page number in function charts</u>
• Setup	819
• Homing	821
• MDI	823
• Automatic mode	826-828
• Roll feed	830

8.1.5 Synchronization

The function chart 831 shows the all the functional blocs which are used in synchronization. Here some of the synchronization functions:

<u>Function name</u>	<u>page number in function charts</u>
• Electronic shaft	
• Electronic gearbox	835
• The change of transmission ratio	
• Electronic cam	839
• The path/angle setpoint	834
• Two interrupt-capable digital inputs	

8.2 Application

Positioning and Synchronization with Virtual Master Axis

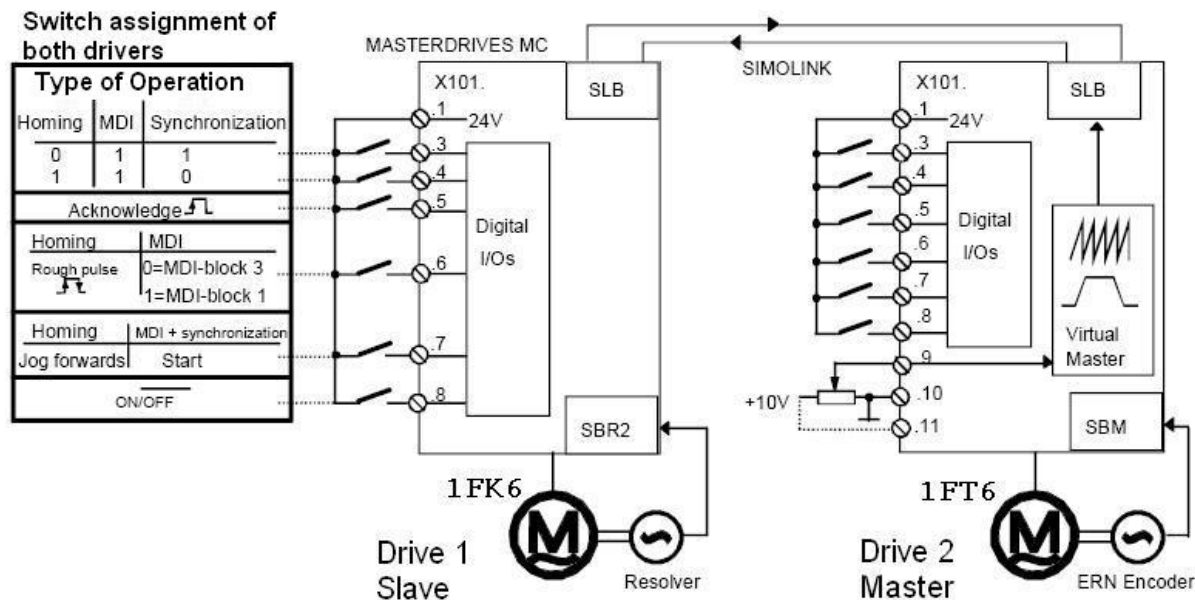


Figure 8.6 Description of the Application

The application includes the following configuration:

- Siemens synchronous servo motors: 1FK6 with resolver and 1FT6 with optical sine/cosine encoder. 1FT6 is driven by the master, which is required for positioning.
- Two MASTERDRIVES MC Converters with technology option F01. Only one of the converters is doing the positioning (Driver 2, the Master).
- Both drives should be operated in the following modes:
 - Homing (this is required for positioning, since resolvers and optical encoders are incremental and not absolute encoders)
 - Point-to-point positioning (MDI; axis type "rotary axis", i.e. without fixed stops)
 - Synchronization with 1:1 transmission ratio using the virtual master axis and the SIMOLINK drive interface
- When the two-axis pack is used, the synchronization can be checked with reference to an LED light beam, which is visible through drilled holes in the flywheel mounted on the motor shafts when the synchronization is operating correctly [4].

SIEMENS Masterdrives MC 6SX7000-0AF10 unit is a training unit, and the positioning had already done by Siemens. The Technology Option F01 had also been enabled. Through Terminal Strip Diagnostics and parameter lists in Drive Monitor one can find out the connection of the Digital Inputs (DI) from 1 to 6 for both drivers.

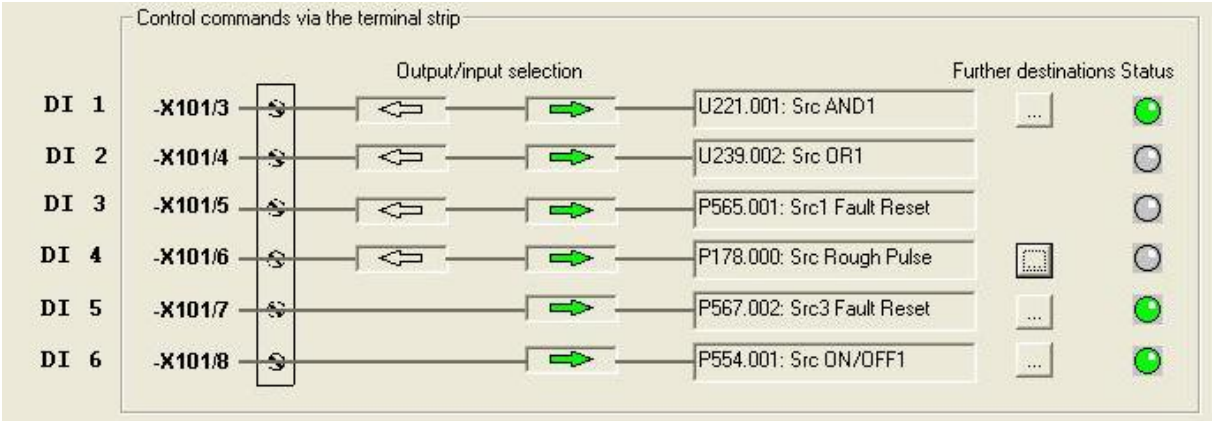


Figure 8.7 Digital Inputs (Terminal Strip Diagnostics)

LEDs show the status of the digital inputs if they are on or off, and the square buttons indicate that there are more than one parameter used to get the input from DIs. If we look at the Figure 3.6 for the status of the DIs; tuning DI-1, DI-2, and DI-3 on results a synchronous operation.

Here are the parameters and interconnections for the manual (using the potentiometer and switches on the unit) operation:

	DI	Parameter #	Index	Value
Mode	DI 1	U221	001	B10 DigIn 1
		U710	029	B10 DigIn 1
	DI 2	U239	002	B12 DigIn 2
Acknowledge	DI 3	P565	001	B14 DigIn 3
Rough pulse MDI no.	DI 4	U178		B16 DigIn 4
		U580	002	B16 DigIn 4
		U536	004	B16 DigIn 4
		U710	010	B16 DigIn 4
		U923	005	B16 DigIn 4
Jog+/ start	DI 5	P567	002	B18 DigIn 5
		U536	005	B18 DigIn 5

		U689		B18 DigIn 5
		U710	003	B18 DigIn 5
		U710	028	B18 DigIn 5
		U923	006	B18 DigIn 5
ON/OFF	DI 6	P554	001	B20 DigIn 6
		555	002	B20 DigIn 6
		U536	006	B20 DigIn 6

Tabelle 8.1 Digital Inputs

8.2.1 Operations modes

8.2.1.1 Synchronous Operation

The below descriptions are true for the both drives, but the only thing should be kept in mind is that; first the master drive (Drive 2), and then the slave drive (Drive 1) should be turned on.

DI-1 = 1	<p>Note: Assuming all DIs are turned off ($DI_x = 0$), and the following the order should be taking place when turning DIs on: DI-1, DI-6, then DI-5. If DI-5 is turned on first the system will not start. In this case Turn DI-5 of then turn DI-6 on. Now DI-3 should be turned on and off again, then turn DI-5 on. The system will now start working. It is possible to change the speed by the potentiometer of the master system.</p>
DI-2 = 0	
DI-3 = 0	
DI-4 = 0	
DI-5 = 1	
DI-6 = 1	

8.2.1.2 Reference run (MDI = 0)

DI-1 = 0	Turn DI-2, DI6, and DI-5 on.
DI-2 = 1	<p>The speed is not changeable by the potentiometers. It is set to 555 rpm for both drivers.</p>
DI-3 = 0	
DI-4 = 0	
DI-5 = 1	
DI-6 = 1	

8.2.1.3 Reference run (MDI = 1)

DI-1 = 0	Turn DI-2, DI-4, DI-6, and DI-5 on.
DI-2 = 1	This time the motors running synchronously at single speed at 27 rpm master, and 69 rpm slave. Here also speed is not changeable by the potentiometers.
DI-3 = 0	
DI-4 = 1	
DI-5 = 1	
DI-6 = 1	

9 Communication with the Masterdrives MC

Communication to the masterdrives is done through the PROFIBUS DP, therefore only this method is going to be discussed. There are other methods can be used to communicate with drivers depending on the application. For more detail on please refer to the compendium. When PROFIBUS DP is used in a high-level automation system, the interface to the masterdrives mc is done by CBP (Communications board PROFIBUS) or CBP2 communications boards.

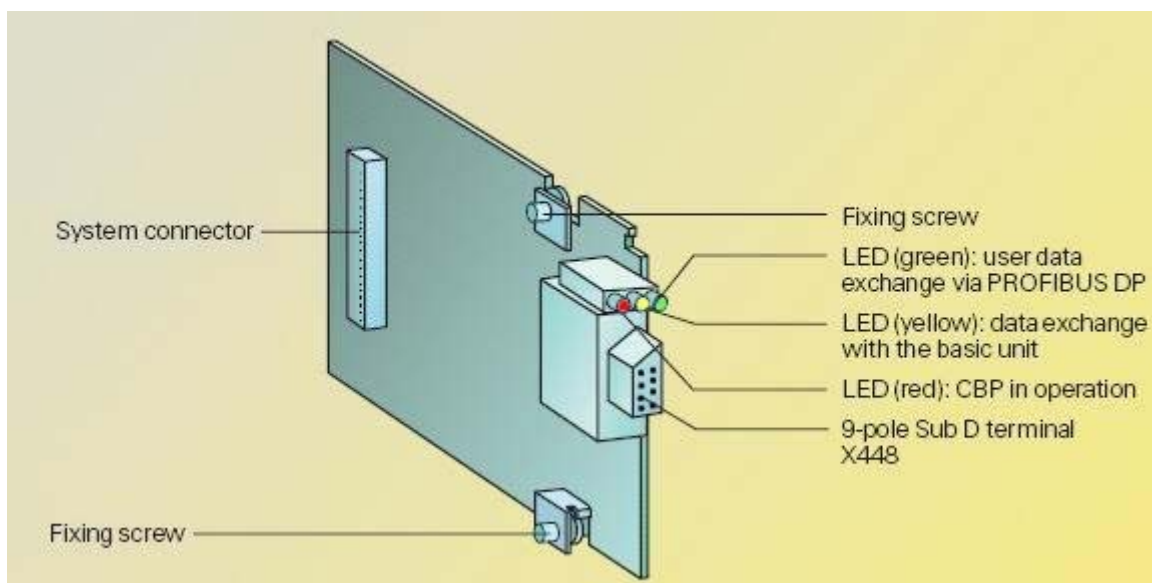


Figure 9.1 CBP communications board (Communications board PROFIBUS)

The communications board has three LEDs, green, yellow, red, provide information on the current operating status. A 9-pole SUB D socket (X448) performs as an interface for PROFIBUS system connection, which complies with the PROFIBUS standard.

All connections of this RS485 interface are short-circuit-proof and floating.

The transmission rate of the CPB is between 9.6 kbaud and 12 Mbaud, which is suitable for optical fibre link. Although it is suitable for an optical fibre link, due to special issues the optical plugs are not installed to the compact Masterdrives MC units.

9.1 Functionality of the CBP

- Cyclical user data exchange with the master according to the “PROFIBUS DP Profile for PROFIDRIVE Variable-Speed Drives ”
- Acyclical communication channel for exchanging parameter values up to a length of 118 words with a SIMATIC S7 CPU.
- Acyclical communication channel for connecting the Drive ES Basic start-up, parameterization and diagnostics tools.
- Support of the PROFIBUS DP control commands, SYNC and FREEZE, for synchronized data transfer from the master to several slaves and vice versa.

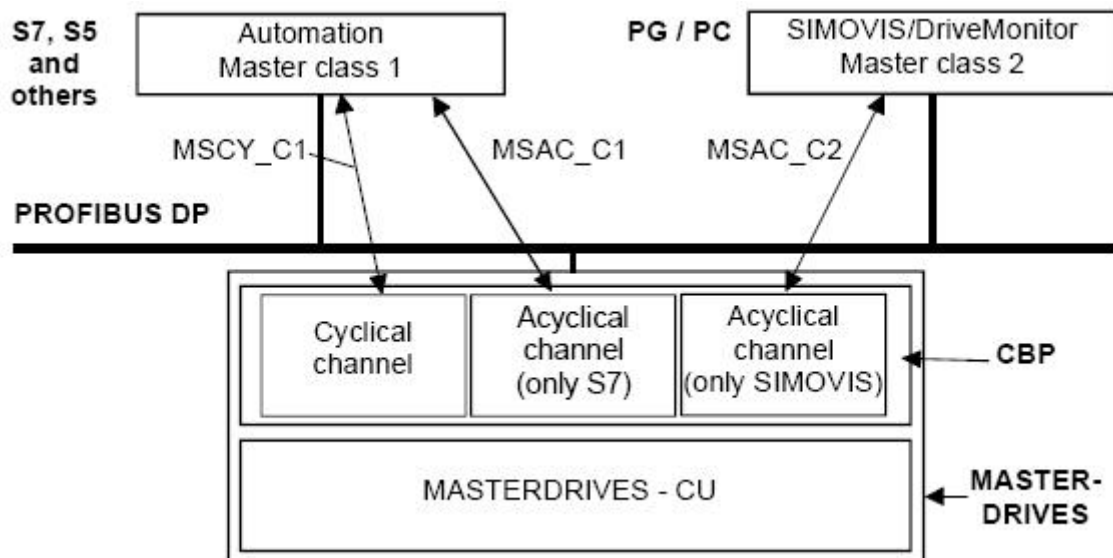


Figure 9.2 Data traffic channels of CBP [4]

9.2 Useful Data

Useful data for the **cyclical MSCY_C1 channel** (see Figure 9.2) is defined in a certain structure in the PROFIBUS profile for variable-speed drives version 2 as a **parameter process data object (PPO)**. Sometimes, the **cyclical MSCY_C1 channel** is called the **STANDARD channel** as well.

The useful data is defined as two parts or areas, which is transmitted in each telegram.

- **Process Data Area (PZD)**; which includes Controlword1 and Controlword2, Statusword1 and Statusword2, Actualvalue, and Mainsetpoint
- **Parameter Area (PKW)**; which is used for reading and writing parameters

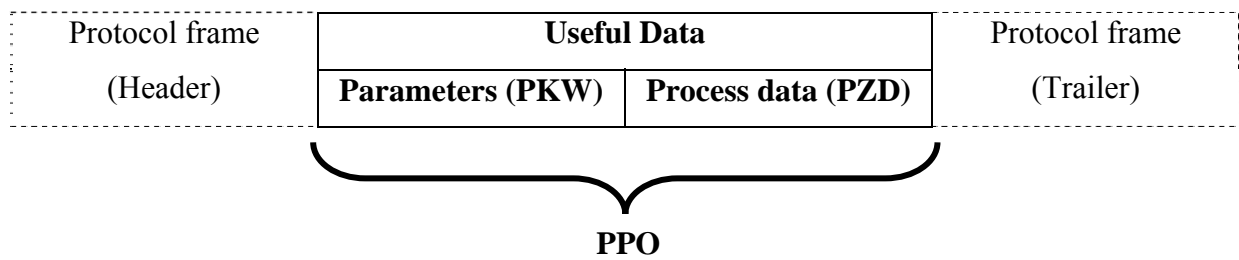


Figure 9.3 Parameter identifier value

There are 5 types of Parameter Process data Objects (PPO). The type of PPO to be used depends on the task. The selection can be done in the S7 Project with the hardware configuration.

	PKW				PZD									
	PKE	IND	PWE		PZD1 STW1 ZSW1	PZD2 HSW HIW	PZD3	PZD4	PZD5	PZD6	PZD7	PZD8	PZD9	PZD10
	1st Word	2nd Word	3rd Word	4th Word	1st Word	2nd Word	3rd Word	4th Word	5th Word	6th Word	7th Word	8th Word	9th Word	10th Word
PPO1														
PPO2														
PPO3														
PPO4														
PPO5														

Figure 9.4 PPO types

- | | | | |
|------|----------------------------|------|-------------------|
| PKW: | Parameter identifier value | STW: | Control word 1 |
| PZD: | Process data | ZSW | Status word 1 |
| PKE: | Parameter ID | HSW: | Main set point |
| IND: | Index | HIW: | Main actual value |
| PWE: | Parameter value | | |

As part of telegrams PPO1, PPO2, and PPO5 one can read and write parameters from the DP slave or to it, by using PKW (parameter identifier value

All of these telegrams PPO1, PPO2, PPO3, PPO4 and PPO5 PZD contain control words and setpoints or status words and actual values.

The type of PPO to be used depends on the application, and they are defined during the hardware configuration. In a configuration for direct data exchange (lateral communication), local input address areas of an intelligent DP slave or of a DP master are assigned to the input address areas of a PROFIBUS-DP partner.

The intelligent DP slave or the DP master uses these assigned input address areas to receive the input data that the PROFIBUS-DP partner sends to its DP master. Depending on the PPO type the number of words in a telegram will vary, and also the address range will vary depending on the PPO type as well.

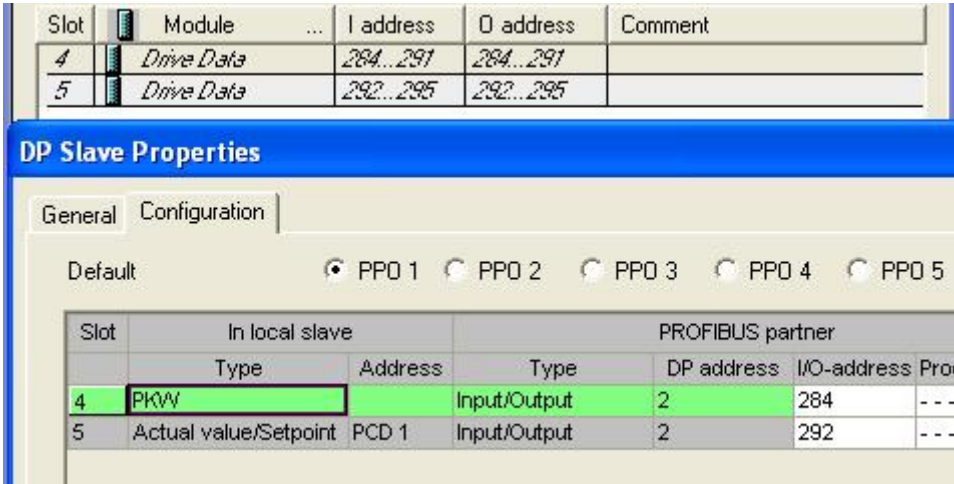


Figure 9.5 PPO overview in hardware configuration

For PPO1 address range is:

- PKW: 284-291 4 words (8 bytes)
- Actual value/setpoint: 292-295 2 words (4 bytes)

For PPO2 address range is:

- PKW: 284-291 4 words (8 bytes)
- Actual value/setpoint: 292-303 6 words (12 bytes)

One can compare the above information with the information given in Figure 9.4 about PPO types to verify that if the actual hardware configuration agrees with the theoretical information.

Master to Slave		PZD1 STW1 1 st Word	PZD2 HSW 2 nd Word	PZD3 3 rd Word	PZD4 4 th Word	PZD5 5 th Word	PZD6 6 th Word	PZD7 7 th Word	PZD8 8 th Word	PZD9 9 th Word	PZD10 10 th Word	
	16 bit process data	3001	3002	3003	3004	3005	3006	3007	3008	3009	3010	
	PPO1, PPO3	PZD2										
	PPO2, PPO4	PZD6										
	PPo5	PZD10										

Slave to Master	PZD1 ZSW 1 1 st Word	PZD2 HIW 2 nd Word	PZD3 3 rd Word	PZD4 4 th Word	PZD5 5 th Word	PZD6 6 th Word	PZD7 7 th Word	PZD8 8 th Word	PZD9 9 th Word	PZD10 0 10 th Word
	P734	P734	P734	P734	P734	P734	P734	P734	P734	P734
	001	002	003	004	004	006	007	008	009	010

Table 9.1 Fixed assignment and combination values

If the technology board is not mounted; one can use the fixed connections shown in Table 9.1. Even if the board is mounted this table is very informative about the wirings of process data from the master to the slave, and vice versa.

	Parameter ID (PKE)															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1 st word	AK				SPM	PNU										
2 nd word	Parameter index (IND)															
	Parameter Value (PWE)															
3 rd word	Parameter Value High (PWE1)															
4 th word	Parameter Value Low (PWE2)															

Table 9.2 Telegram Structure PKE area

The Table 9.2 only complies with the PPO1, PPO2, and PPO5. By using one of these PPOs one can write or modify parameter in DP slaves. Normally parameterization is done through Drive monitor or OP1S, but in S7 Project it is important to initialize or change some of the

parameters depending on the application. For that reason PKW part of PPO is used very often in the program. Figure 9.6 shows a piece of code which makes use of system function (SFC 15 "DPWR_DAT") to send a telegram which includes the information in the format of Table 9.2.

Name	Data Type	Address	Initial Value	Exclusion address
word1	Word	0.0	W#16#0	<input type="checkbox"/>
word2	Word	2.0	W#16#0	<input type="checkbox"/>
word3	Word	4.0	W#16#0	<input type="checkbox"/>
word4	Word	6.0	W#16#0	<input type="checkbox"/>
DB_Address	Word	8.0	W#16#0	<input type="checkbox"/>

```

FBI : Write
Comment:

Network 1: Write Parameters to DP Slave
Comment:

CALL "DPWR_DAT"           // Write consistent data to DP Slave
LADDR :=#DB_Address       // DP Slave address e.g.: 272
RECORD :=P#L 0.0 BYTE 8   // 8 bytes of data including word1 to word4
RET_VAL:=#Alarms         // Alarms to be stored if occurred
  
```

Figure 9.6 SFC 15 writing data to DP sale

Table 9.2 shows the structure of the PKE of a telegram. Where AK: Task ID or reply ID, SPM: Toggle bit for processing the parameter change report, PNU: Parameter number.

Task ID	Significance	Reply ID	
		Positive	Negative
0	No task	0	7 or 8
1	Request parameter value	1 or 2	7 or 8
2	Change parameter value (word)	1	7 or 8
3	Change parameter value (double word)	2	7 or 8
4	Request description element	3	7 or 8
5	description element (not with CBP)	3	7 or 8
6	Request parameter value (array)	4 or 5	7 or 8
7	Change parameter value (array, word)	4	7 or 8
8	Change parameter value (array, double word)	5	7 or 8
9	Request the number of array elements	6	7 or 8

Table 9.3 Task ID master to converter

The task ID 7 is the most commonly used one.

Example:	Decimal	Hex
PNU (Parameter):	554	22A
IN (Index):	001	1
AK (Task ID):	7	7
PWE (Parameter value):	3100	C1C
SPM:	0	0

From table 4.3 the telegram can be constructed as;

1 st word:	AK	SPM	PNU:	722A
2 nd word:	IN			001
3 rd word:	PWE1			0000
4 th word:	PWE2			C1C

9.3 Starting the DP Slave through S7-Project

During the hardware configuration the DP Slave PROFIBUS bus addresses were configured. DP Slave is already connected to the Drive Monitor, and it is possible to parameterize it manually with Drive Monitor. Before starting to write the control program (S7 Program), we are going to test the connection with SIMATIC S7-300, and run it through diagnostics tool of S7-Project.

To do that; first some interconnections have to be done by connection some connector to certain parameter so that it will be possible to gain the control over the control words and Mainsetpoints. When the MASTSTERDRIVES MC training unit is turned on; by default it will listen to the digital inputs from the manual switches. In order to control the unit from the S7-project by using PROFIBUS connection; DIs must be taken from control words. For that each DI should be addressed to the relevant bit of the control word. Binectors of the control word1 are listed in Figure 9.7. For example bit number 0 is connected to the binector B3100. For rest of the binector connections please see Table 9.4.

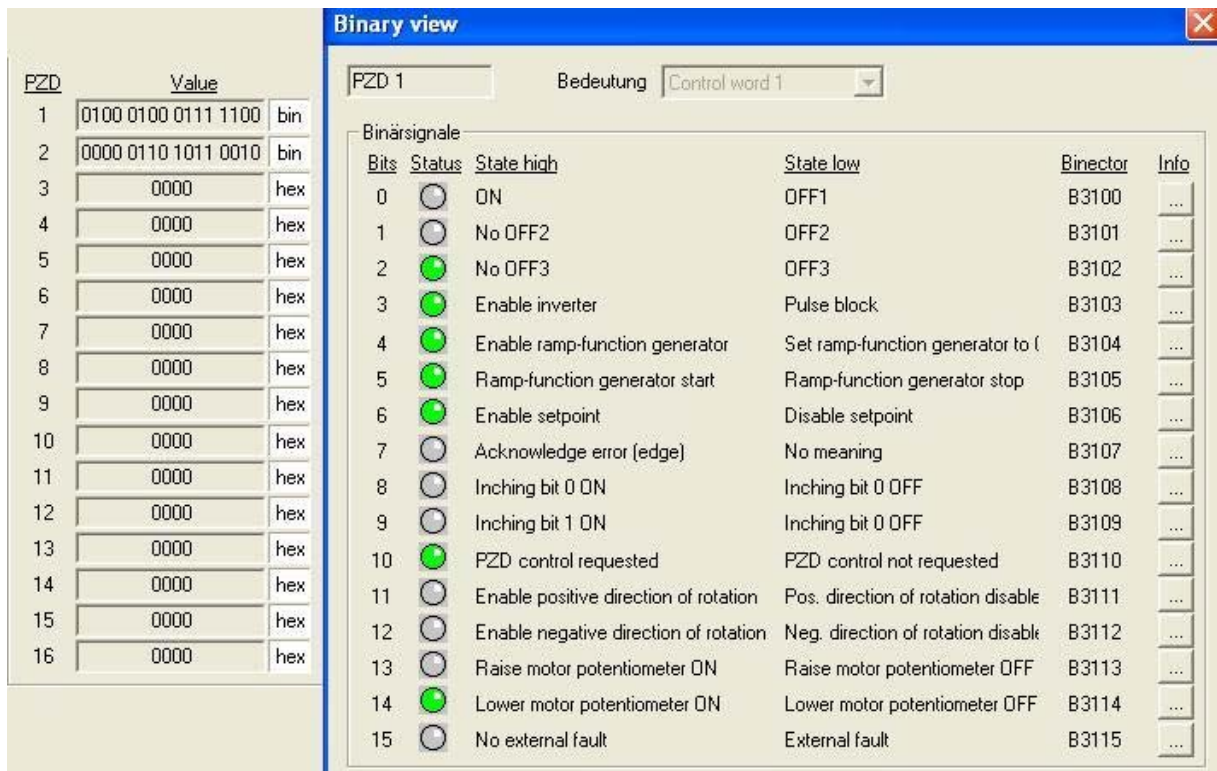


Figure 9.7 PZD1 (Control word 1)

	DI	Parameter #	Index	Value	
Mode	DI 1	U221	001	B3100	
		U710	029	B3100	
Acknowledge	DI 2	U239	002	B3101	
Rough pulse MDI no.	DI 3	P565	001	B3102	
		DI 4	U178		B3103
			U580	002	B3103
			U536	004	B3103
			U710	010	B3103
U923	005		B3103		
Jog+/ start	DI 5	P567	002	B3104	
		U536	005	B3104	
		U689		B3104	
		U710	003	B3104	
		U710	028	B3104	
		U923	006	B3104	
ON/OFF	DI 6	P554	001	B3105	
		555	002	B3105	
		U536	006	B3105	

Table 9.4 Connection DI to PZD1

Once all the parameters modified according to the Table 9.4, there is one more thing to do; which is connecting the mainsetpoint (KK3032) to the parameter U681 (2681).

Now it is possible to start/stop and change the speed from the diagnostic tool in S7 project.

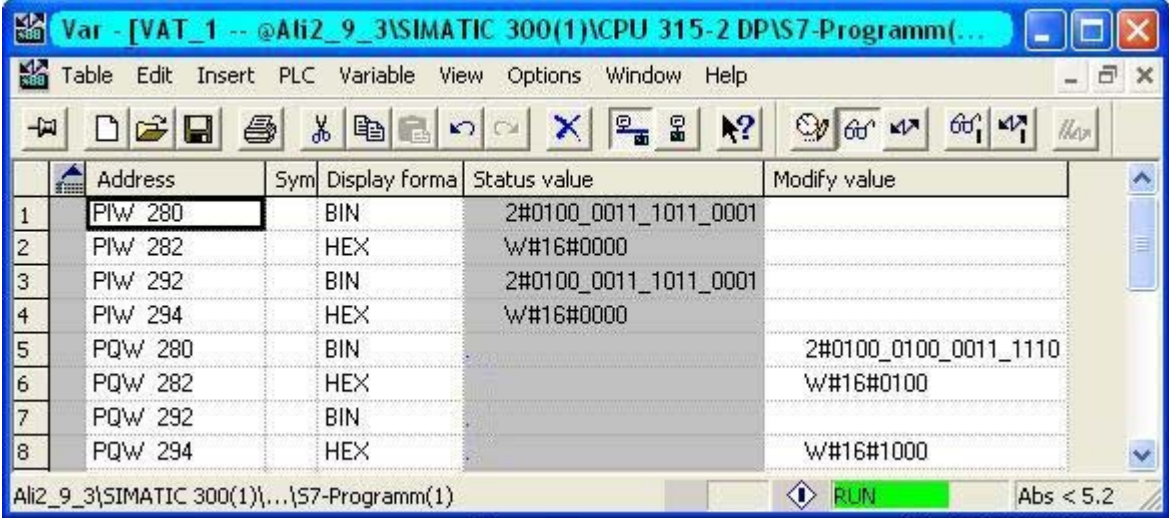


Figure 9.8 Diagnostics tool read/write status and control words

Here are the commands for reading and writing PZD of the driver from SIMATIC S7-300;

Read command	Address	Write command	Address
PIW	280	PQW	280

PIW 280: read status word1 at the address of 280 in DP slave

PQW 280: write the control word1 to the address of 280 in DP slave

10 S7- Programming and Control

The control program is written in STEP 7 as Statement List (STL), which is a textual programming language that can be used to create the code section of logic blocks. STL syntax statements are similar to assembler language and consist of instructions followed by addresses on which the instructions act. STL programming is very similar to machine languages; it has all the necessary elements for creating a complete user program. It contains a comprehensive range of instructions, and a total of over 130 different basic instructions and a wide range of addresses are available. There are functions, function blocks, and data blocks allow you to structure your STL program. It is not intend of this chapter to go through all the code of program, but explain some of the key functionalities.

These are some documents one can read when/if needs some help;

- Working with STEP 7 V5.3, Getting Started Manual
- Programming with STEP 7 V5.3
- Configuring Hardware and Communication Connections, STEP 7 V5.3
- From S5 to S7, Converter Manual

The order number of the above documents is: **6ES7810-4CA07-8BW0**

Block	Description
Organization Blocks (OB)	OBs determine the structure of the user program.
System Function Block (SFC)	SFBs and SFCs are integrated in the S7 CPU, and permit the programmers access to some important system functions.
Function Block (FB)	FBs are blocks with a "memory" which can be programmed by the programmer.
Functions (FC)	FCs contain program routines for frequently used functions
Instance Function Blocks (Instance DB)	Instance DBs are associated with the blocks when FBs of SFBs are called. They are created automatically during compilation.
Data Blocks (DB)	DBs are data areas for storing user data. In addition to the data that are assigned to a function block, shared data can also be defined and used by any blocks.

Table 10.1 Block types

The control program consist of three main sections and a subsection; as they are seen in the Figure 10.1; Synchronous operation, Asynchronous operation, Reset Synchronous operation, Reset Asynchronous operation, which are communicating between Human Machine Interface (HMI) and Masterdrives MCs. There sub section User Friendly HMI, designed to run synchronous operation with more self descriptive GUI. User Friendly HMI will be an optional HMI in Synchronous operation’s page. It uses the Synchronous operation mode’s functionalities; therefore it will not be explained separately.

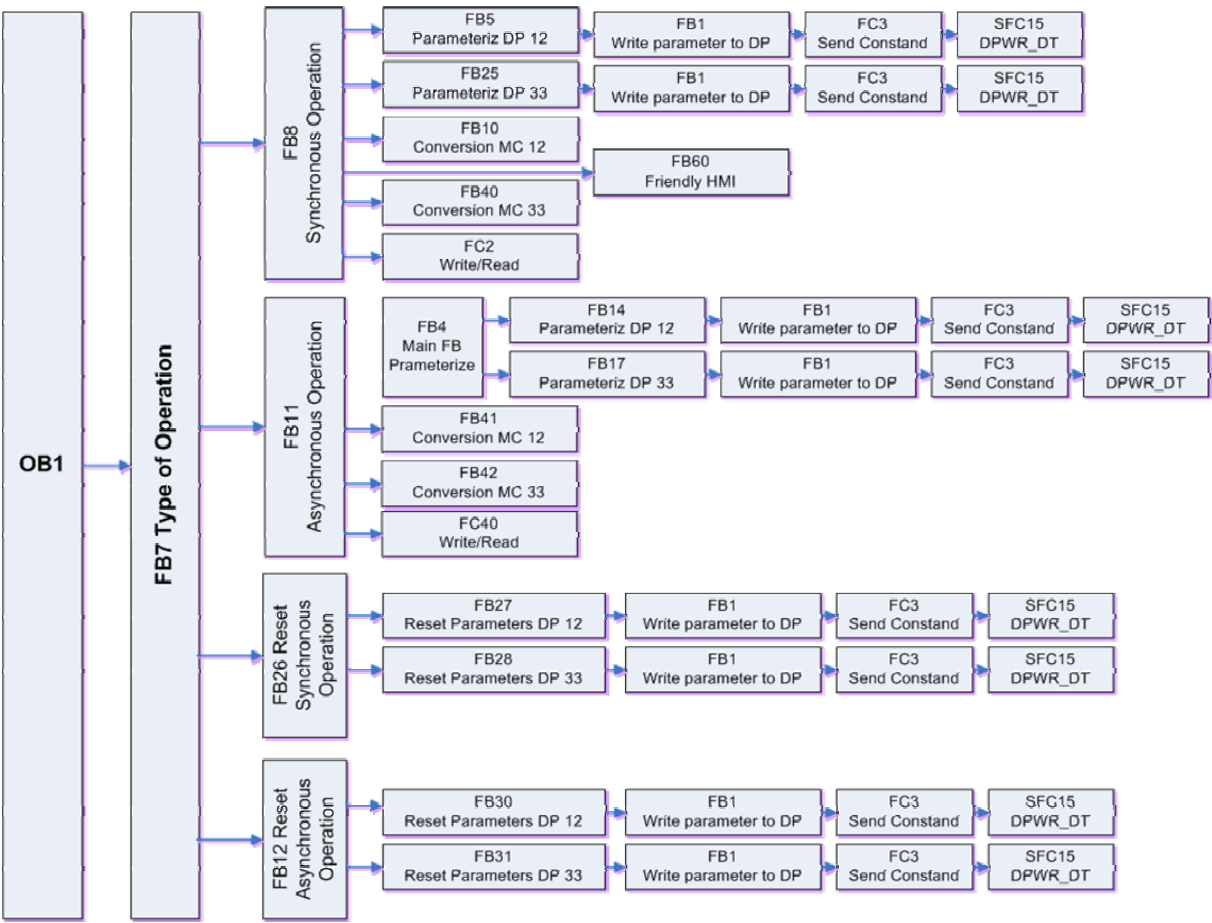


Figure 10.1 Flow diagram of S7 Control program

As program starts at first the type of operation must be decided. Depending on the operation new user interface will show up. If asynchronous operation is chosen then it is ready for setting the speed independently, if synchronous operation is chosen then the next step is to decide the mode of the operation, and if this user interface is complicated for you, then you may want to choose the “User Friendly HMI”, which is easier to operate. There are three modes that can be chosen from; synchronous mode, reference mode and MDI.

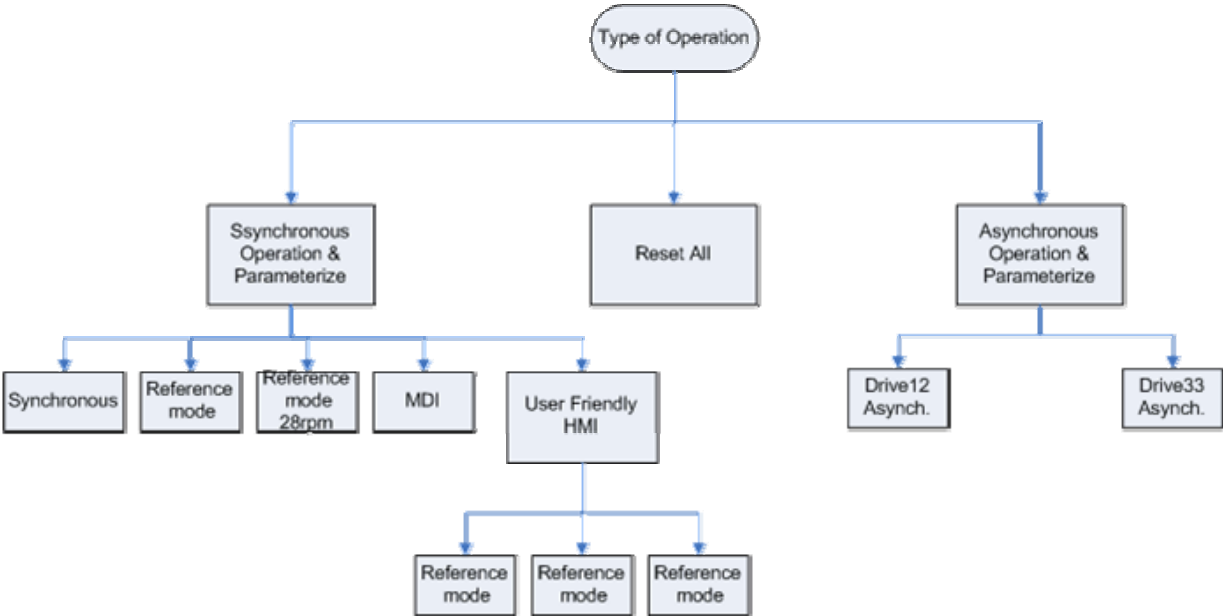


Figure 10.2 Flow diagram of control program

10.1 Synchronous Operations

Points of programming view, both operations (synchronous and asynchronous) have some commonalities that can be expressed by the flow diagram.

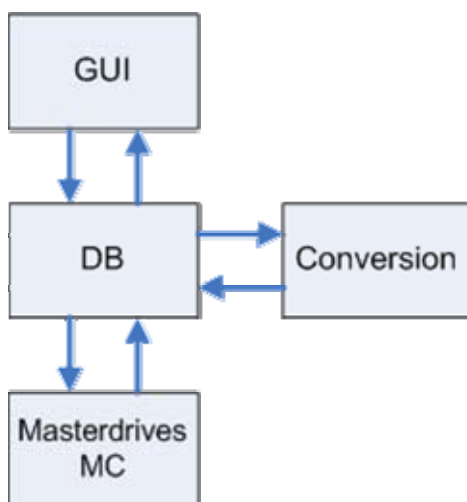


Figure 10.3 Common program flow

10.1.1 Parameterization

In order to bring the Masterdrives MCs in the state that they are ready for the any given type of operation, they must be parameterized. Just because there are more than one parameter to configure there must be a piece of code that repeats the write operation in a loop, where the number of parameters defines number of iterations.

1. Word	Bit-Nr.	15	12	11	10	0
		AK		SPM	PNU	
2. Word	Bit-Nr.	15	8		7	0
		Index High			Index Low	
3. Word		Parameter Value High			PWE1	
4. Word		Parameter Value Low			PWE2	

Table 10.2 Telegram PKW area

To change (to write a value to a parameter in DP slave); one must use the system function called SFC15, which only accepts the telegram in a certain format which is described in Table 10.2.

SFC15 “DPWR_DAT” :

```

CALL "DPWR_DAT"           //SFC15
  LADDR :=#Adress_EA      //The address of DB slave
  RECORD :=P#L 0.0 BYTE 8 //The prepared telegram to be sent
  RET_VAL:=#Alarm_send    //Alarm
  NOP 0
  
```

The Figure 10.4 is the flow diagram is a section from the S7 Program, which performs the parameterization task every time it is called. This section of the code is used for two main tasks parameterizing the Masterdrives MCs, and resetting them. The reset task is basically same as parameterization but the values are the default values that were before the change took place.

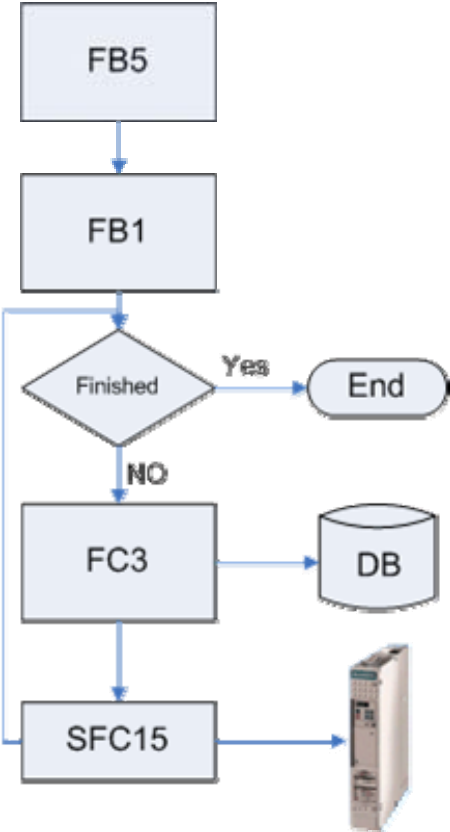


Figure 10.4 Parameterization flow diagram

Let us look at the function blocks used in Figure 10.4;

FB5:

In FB5 the DB number is assigned to the variable called “DB_Number”, to let the function know which database is used for the data to be fetched from, and let the function know that how many times it must repeat the task. Here is the code:

```
L 20
T #Iteration // number of times hat loop take place
L 85
T #DB_Number // DB 85 is going to be used
CALL FB1, DB1
Iteration:=#Iteration
DB_Number:=#DB_Number
```

FB1:

FB1 makes sure that after every read operation it points to the next parameter. Each parameter takes 4-word space in database so the pointer has to jump 4 words after every read operation. The only exception is that the first read operations start at the second line of the DB, because the first line is the address of the Masterdrive MC (e.g.:272)

```
L 2 // jump to the next line where the first parameter is
T #JumpAddress_DB
L #Iteration // Number of iterations for loop
next: T MB 10
CALL FC3
DB_Number :=#DB_Number
JumpValue :=#JumpAddress_DB
Parameter_Sent:=#Parameter_Sent
L #JumpAddress_DB
L 8
+l // add 8 bytes after each reading of parameter
T #JumpAddress_DB
L MB 10
LOOP next
```


Here is the small portion of a database, where the data is been read from.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	PKW_Address	INT	272	PKW start address
+2.0	DI_PNU	INT	567	DI 5 Satarts from here
+4.0	DI_Index	INT	2	
+6.0	DI_AK	INT	7	
+8.0	DI_PWE	WORD	W#16#3104	
+10.0	DI_PNU1	INT	2536	Parameter number
+12.0	DI_Index1	INT	5	Index
+14.0	DI_AK1	INT	7	Task ID
+16.0	DI_PWE1	WORD	W#16#3104	Parameter Value

Figure 10.5 DB85 Database used for MC 12

FC3:

FC3 does the main job; it constructs the telegram in format seen in [Table 10.2](#) and sends it using the SFC15 to Masterdrives MC.

Word 1 & Word 2

```

L   #DB_Number           //Load DB number
T   #DB_Number_TEMP     //and save it
OPN DB [#DB_Number_TEMP] //Open Database with the corresponding DB#
L   #JumpValue          //Decide where to start reading
SLD 3
LAR1
L   DBW [AR1,P#0.0]     //Read the relevant parameter
T   #Parameter_INT     //Save the parameter value
L   2000                //if the value is bigger than 2000
>=D
JCN M001
L   #Parameter_INT
L   2000
-D                          //is it a U-Parameter
T   #Parameter_INT
L   W#16#80
T   #PNU_biger_2000
M001: L #JumpValue      //Decide which value to read
L   2                    //load 2 in Accu 1

```

```

+i                                     //increase jump value by two
T #Jump_Mark_2Byte_Further //transfer to jump in db
SLW 3
LAR1
L DBW [AR1,P#0.0] //load a word from the DBs
SLW 8 //Shift Index 8 Bit to the left
L #PNU_biger_2000
+D
T #INDEX_Word_2 //Save the value as word 2
L #Jump_Mark_2Byte_Further //find out which bit to read
L 2
+i                                     //increase the jump value
T #Jump_Mark_2Byte_Further //decide which bit to read
SLW 3
LAR1
L DBW [AR1,P#0.0] //load a word from the DBs
T #AK_Save //Save the value as Task ID
L #Parameter_INT //Parameter number in Accu 2
L #AK_Save //Task ID in Accu 1
SLW 12
OW
T #AK_PNU_Word_1

```

Word 3

```

L 0
T #PWE_Word_3

```

Word 4

```

L #Jump_Mark_2Byte_Further //jump value
L 2
+i                                     //jump value increased by 2
T #Jump_Mark_2Byte_Further
SLW 3

```

```

LAR1
L   DBW [AR1,P#0.0]           //load a word from the DBs
T   #PWE_Word_4              //save it as word 4

```

Read the address of Masterdrive DM:

In every database the first line is the address off the DP slave, for example first line of DB85 is 272, which belongs to Masterdrives MC 12

```

L   DBW  0                    //The first line of the DB is address of DP slave
T   #Adress_EA                //PKW Address as Word for SFC15

```

SFC15:

SFC15 is a system function in SIMATIC S7 300, which is ready to use by the programmers. It can be used two different ways; either with direct addressing or indirect addressing. It is always easy to do tests or diagnostics with direct addressing. The only thing to be kept in mind is that the numbers should be converted to Hexadecimal format.

CALL "DPWR_DAT"

```

LADDR :=#Adress_EA           //The address of DB slave
RECORD :=P#L 0.0 BYTE 8     //The prepared telegram to be sent
RET_VAL:=#Alarm_send        //Alarm
NOP 0

```

10.1.2FB10 Conversion

There are few tasks done in FB10;

1. Percentile speed <==> decimal speed: the Masterdrives MC takes the speed in percentile for example 4000 is equivalent to 100% and 2000 corresponds to 50% of the full speed. The reason for that is; the values stored in the connectors are normalized values, with a few exceptions (e.g. connectors for control words). The value range of these connectors covers a percentage value range of:

-200 % (8000H) to +199.99 % (7FFFH).

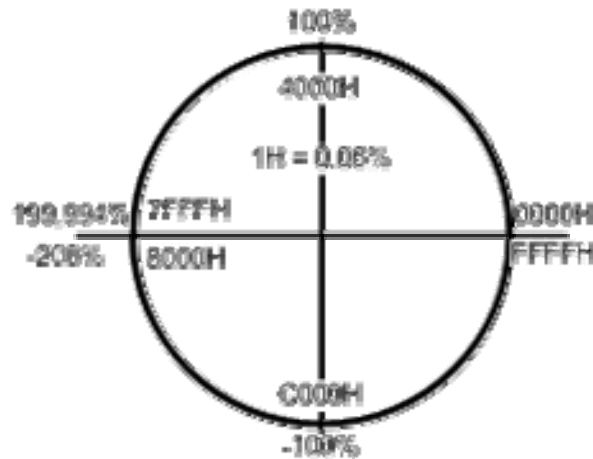


Figure 10.6 value range of 16-bit connectors (Kxxxx) [1]

According to Figure 10.6 one can calculate the constant value transfers Hex value to the decimal value, keeping the maximum speed of the motor in mind the theoretical value should be;

$$\begin{aligned}
 6000 \text{ rpm} &= 4000\text{H} = 16384\text{Dec} \\
 1 \text{ rpm} &= K \\
 K &= 16384/6000 = 2.730666
 \end{aligned}$$

Which means when we want to set speed to 300rpm then we must multiply this value with $K = 2.730666$. Although this value does not work, because the application saved in the Masterdrivers MC limits the maximum motor speed to 2730rpm.

$$\begin{aligned}
 2730 \text{ rpm} &= 4000\text{H} = 16384\text{Dec} \\
 1 \text{ rpm} &= K \\
 K &= \frac{16384}{2730} = 6
 \end{aligned}$$

When $K = 6$ is used the entered speed is equal to the speed read in PMU of the driver.

2. Check if entered speed exceeds the limit (SetValue>2500rpm):

To make sure to not overdrive the motors the speed is verified against a maximum value. With the manual operation of the unit the maximum speed is about 2730rpm, therefore to be on safe side I set the maximum speed limit to 2500rpm. If the maximum limit is exceeded the value is not sent to Masterdrive MC, instead a warning bit is set (SetValueTooBig), which alerts GUI to display a message that the entered value is too large.

3. Initialize some of the bits in Control word:

Some of the bits in control word can be set when at the beginning of the operation. For example the bit 11 is making the system to listen the PROFIBUS.

10.1.3 FC2 Write & Read operations for Control and Status words

There are external input and external output addresses and S7 functions to access these addresses.

The screenshot shows two configuration windows for masterdrives. The first window is for masterdrive (12) and the second is for masterdrive (32). Both windows show a table with columns for Slot, Module, I address, O address, and Comment. In both cases, slots 4 and 5 are configured as Drive Data modules with specific address ranges.

(12) MASTERDRIVES CBP					
Slot	Module	...	I address	O address	Comment
4	Drive Data		272...279	272...279	
5	Drive Data		280...283	280...283	

(32) MASTERDRIVES CBP(1)					
Slot	Module	...	I address	O address	Comment
4	Drive Data		284...291	284...291	
5	Drive Data		292...295	292...295	

Figure 10.7 Input and Output addresses of peripherals

As it is explained before; peripheral addresses were configured during the hardware configuration (see Appendix 1). Depending on the PPO types address range will change. For example PPO1 has the 4+2 word format. The first four words describe PKW area and 2 words describe PZD area. Here are the addresses of the both drivers 12, and 32, shown in Table 10.3. This table is an interpretation of Figure 10.7, where the address range of the parameter ID value (PKW), and process data (PZD) are shown. Figure 10.7 is screen shots from hardware configuration.

Masterdrive MC	Useful Data					
	Parameters (PKW)				Process data (PZD)	
	Word 1	Word 2	Word 3	Word 4	PZD1 STW1 ZSW1	PZD2 HSW HIW
12	272	274	276	279	280	282
32	284	286	288	290	292	294

Table 10.3 PKW and PZD peripheral addresses

Name of Area	Function of Area	Access to Area via Units of the Following Size:	Abbrev.
I/O: external input	This area enables your program to have direct access to input and output modules (that is, peripheral inputs and outputs).	Peripheral input byte	PIB
		Peripheral input word	PIW
		Peripheral input double word	PID
I/O: external output		Peripheral output byte	PQB
		Peripheral output word	PQW
		Peripheral output double word	PQD

Table 10.4 S7 functions to write and read external input and outputs

Here is a piece of code from FC2, which is used for reading (PIW) actual from the address of 282 at the peripheral side (first two lines). Next two lines of the code write (PQW) control word to the peripheral.

```

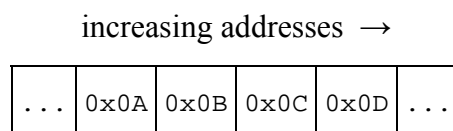
L   PIW 282                //Read Actual Value (input value)
T   "DB10 for DB100 interface".ActualValue //Save it to DB10

L   DB100.DBW 9            // Take Control word from DB100
T   PQW 280                //Write Control word to the DP
                                // slave (output value)

L   "DB10 for DB100 interface".SetValue    // Take the setpoint value from
                                                //DB40
T   PQW 280                //Write setpoint

```

Control word is named “ControlWord” and defined as word in DB100. The SIMATIC S7-300 has a **big-endian** memory structure, which means for example With 8-bit atomic element size and 1-byte (octet) address increment.



Here is the control word address range in database. For any kind of interaction with the Masterdrives MCs these address of each bit must be known so that it is possible to set bits to “0” or “1”.

Control word								
Address	Bits from 0-7							
9.0	9.7	9.6	9.5	9.4	9.3	9.2	9.1	9.0
10.0	10.7	10.6	10.5	10.4	10.3	10.2	10.1	10.0

Using the **big-endian** structure control word become as seen below

Control word															
Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
9.7	9.6	9.5	9.4	9.3	9.2	9.1	9.0	10.7	10.6	10.5	10.4	10.3	10.2	10.1	10.0

Table 10.5 Control word in big-endian structure

Now it is easy to assign DIs to the control word. In this project for synchronous operation DI assignments are as follows;

Digital Input (DI)	Significance	Bit address in DB100
DI-1	Operation mode	10.0
DI-2	Operation mode	10.1
DI-3	Acknowledge	10.2
DI-4	Reference operations	10.3
DI-5	Start/Stop	10.4
DI-6	ON/OFF	10.5

Table 10.6 DI assignments

Digital Input wirings also done with HMI to DB100 as well. That is how the connection between the HMI and the S7 program interims of control word is done.

10.1.4FB60 User Friendly HMI

The User Friendly HMI function block is making use of all the previous steps, except it combines some of the control bits under one buttons function so that the user will be able to run a specific mode just pressing one button for example DI-1 for driver-1 and driver-2 are controlled by one button called “Synchronous”.

This is an optional module, and does not add a new functionality to the operation modes.

10.2 Asynchronous Operation

As it is seen from Figure 10.1, in terms of programming point of view, asynchronous operation is very similar to synchronous operation. The difference between two operation types is interconnection of connectors and binectors, which results using a different set of parameters and parameter values during parameterization.

Control word in Table 10.5 is used for asynchronous operation as well. Each bit is assigned to the parameters shown in Figure 10.8, and the assignment of the binectors from 3100 to 3114 done by parameterization process, where the parameters are saved in DB15.

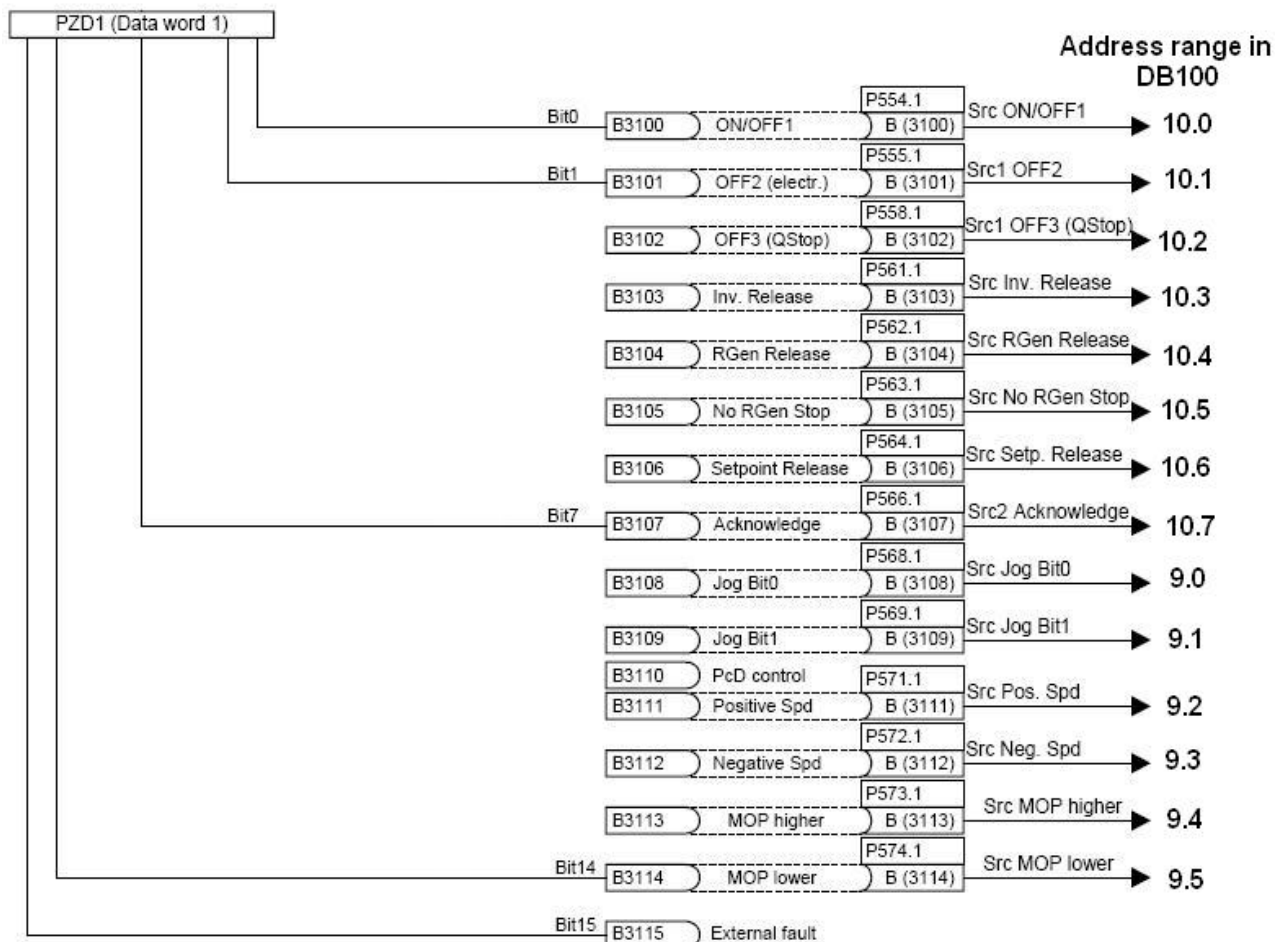


Figure 10.8 PZD1 Control word 1

The speed set point (KK3032) is connected to P209. The speed actual value (KK91) is connected to P734.2.

In asynchronous operation there is no master slave relationship between two Masterdrives MC therefore both drivers operate independently with any speed and direction.

10.3 Reset Functionality

To switch from one operation type to other; some of the function blocks must be re arranged. Functions used one operation may not be used in another so, it is a good idea to bring the Masterdrivers MCs in to initial state, and then re-parameterize them accordingly. To bring parameter values to original state a reverse parameterization is done. It is done exactly the way normal parameterization would be done, but the values of the parameters are the initial values saved before any change.

Besides resetting parameter the control word also reset so that none of the operation modes are in run mode.

11 Human Machine Interface (HMI)

The user interface means that people (users) interact with a particular machine, device, computer program or other complex tool, which provides means of; **Input**, allowing the users to manipulate a system, **Output**, allowing the system to produce the effects of the users' manipulation.

The main types of HMI are;

- **Command-line interfaces:** It is the oldest and a most powerful user interface, but the only problem is that; it is not user friendly, because user must remember needed commands. Input: text command typed by the user. Output: printed text on the monitor by the system.
- **Graphical user interface (GUI):** The system accepts input through devices such as keyboard and mouse and provide articulated graphical output on the computer monitor. The user interface usually looks like a real device that user already has some idea how to use. For example a soft multimedia player can operated via an interface with buttons, switches, windows, which looks like a radio or a recorder.
- **Web-based user interfaces:** They accept input and provide output by generating web pages which are transported via the Internet and viewed by the user using a web browser program. Newer implementations utilize Java, AJAX, Microsoft.NET, or similar technologies to provide real-time control in a separate program, eliminating the need to refresh a traditional HTML based web browser [8].
- **Touch interfaces:** Touch interfaces are also using graphical user interfaces on touchscreen displays. The screen is used for inputs and outputs as well. This type of HMI is used in many types of industrial processes and machines.

In this project the second type of HMI (GUI) is implemented in conjunction with the third type (Web-based user interface). A separate real-time Java and S7-Program used to manipulate the data. Via java beans as java applets HMI is displayed on a web page for taking inputs and printing the outputs. To do java programming one can use different editors like Borland JBuilder, UltraEdit, and Eclipse and so on. Eclipse is an open-source software, which comes for no cost. It is also one of the most used editors as well. There are a lot of examples, plugins available on the Internet. Therefore in this project Eclipse is chosen to develop the HMI, in java programming.

11.1 Eclipse



Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. A large and vibrant ecosystem of major technology vendors, innovative start-ups, universities, research institutions and individuals extend, complement and support the Eclipse platform [9].

Eclipse is not only a java editor, today there are tens of projects been developed by its open source community, and the project areas can be categorized in about 7 groups;

1. Enterprise Development
2. Embedded and Device Development
3. Rich Client Platform
4. Rich Internet Applications
5. Application Frameworks
6. Application Lifecycle Management (ALM)
7. Service Oriented Architecture (SOA)

Eclipse does not come with a Java virtual machine (JVM), so you have to get one yourself. Note that Eclipse 3.0 needs a 1.4-compatible Java runtime environment (JRE).

To use Eclipse effectively, you will need to learn how to make Eclipse use a specific JRE. In addition, you may want to influence how much heap Eclipse may allocate, where it loads and saves its workspace from, and how you can add more plug-ins to your Eclipse installation. Of course one does not need every plugin available. They must be chosen depending on need for the specific projects. For example for the this project a plugin isstalled to save the project as a Jar file, which reduces the saved project as a compressed single file.

11.1.1The Eclipse's licence

Eclipse uses the Eclipse Public License (EPL). The EPL is a commercially friendly license that allows organizations to include Eclipse software in their commercial products, while at the same time asking those who create derivative works of EPL code to contribute back to the community.

11.1.2 How start using Eclipse

In order to get Eclipse running a few components have to be downloaded from the Internet and installed;

- A compatible Java runtime environment (JRE)
- Eclipse Software Development Kit (SDK)
- Plugins that are required for specific purposes
- Java Swing a GUI tool kit for java

There are also a lot of tutorials, books and white papers to get more information and help from. In addition, newsgroups are another useful way to find solutions to certain problems.

11.1.3 Download the JAVA S7 Beans Library from SIEMENS site

In automation HMI development in Java with SIMATIC requires Java S7 beans, where they contain some of the soft gadgets such tachometer, level indicators, and thermometers. They also enable java program to communicate with communication processor Siemens CP 343-2 IT device.

The library can be downloaded from siemens' internet site;

<http://support.automation.siemens.com>


Automation and Drivers Service & Support → Navigation → Communication/Networks→ SIMATIC NET Industrial Communication→ Industrial Ethernet → System interfacing→ SIMATIC S7→ CP 343-1 Advanced

Here are the files to be downloaded:

- S7BeansAPI Release V2.5.5.zip (266 KB)
- S7BeansAPI JAVADocu V2.5.5.zip (116 KB)
- S7BeansAPI Runtime V2.5.5.zip (196 KB) As a result of code optimization and thus quicker loading times, a runtime library without debug code is additionally available. It can be loaded to the file system of the CP. (directory: /applets).

11.1.4Java visual project

Step by step a java project will be described to start designing a GUI.

- Start Eclipse by double clicking on  icon
- When select a work space window comes up; select a specific place where you want to save your current project.
- When Eclipse is open, go to menu choose file→New→Project→Java Project

Name the project (e.g.: Thesis_HMI) →next

You will have a window with few tabs, as shown blow;

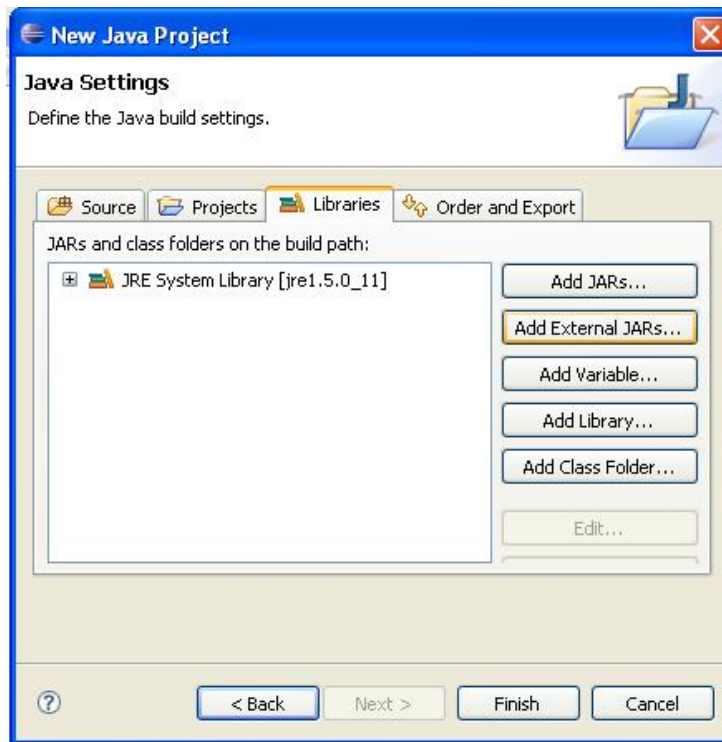


Figure 11.1 Add External JARs

- Choose Libraries tab → Add External JARs

Now browse file selection window where the Siemens' S7 APIs are saved (they had to be unzipped). Select four of the APIs (s7api.jar, s7applets.jar, s7gui.jar, s7uti.jar), and press open button.

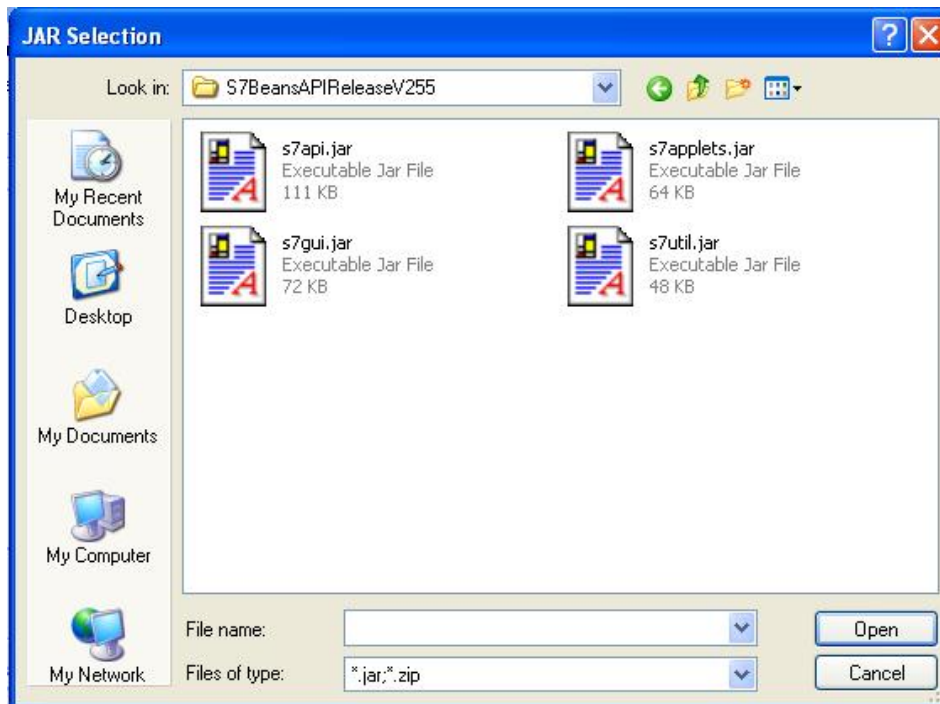


Figure 11.2 Import External JARs to the project

- → finish
- Now the Eclipse window should look like;

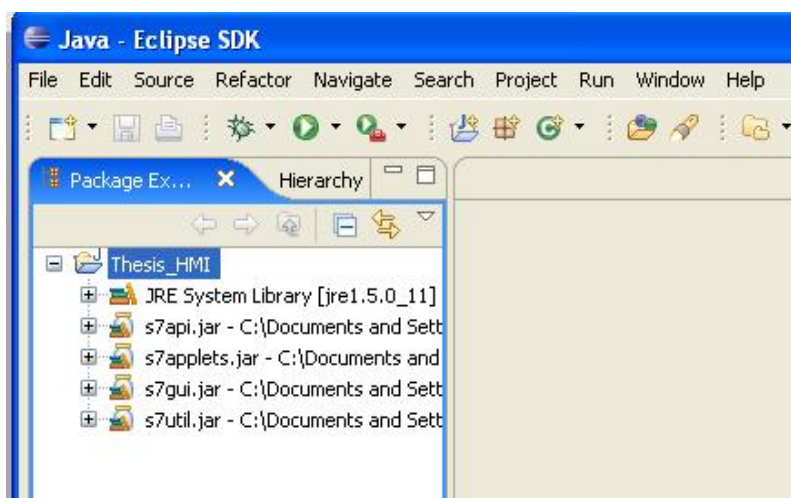


Figure 11.3 The project window with APIs

- to add visual class; → file → new → Visual Class

Now we are ready to program the HMI using Siemens Java S7-Beans. To use S7-Beans in the program one need to understand the mechanics of the java APIs. The next step is to explain the S7-Beans APIs.

11.2 S7-Beans and interconnection hierarchy

Creating the Java applet is based on the program library in S7 beans APIs, which provide for the communication between the Java user program and the S7 program.

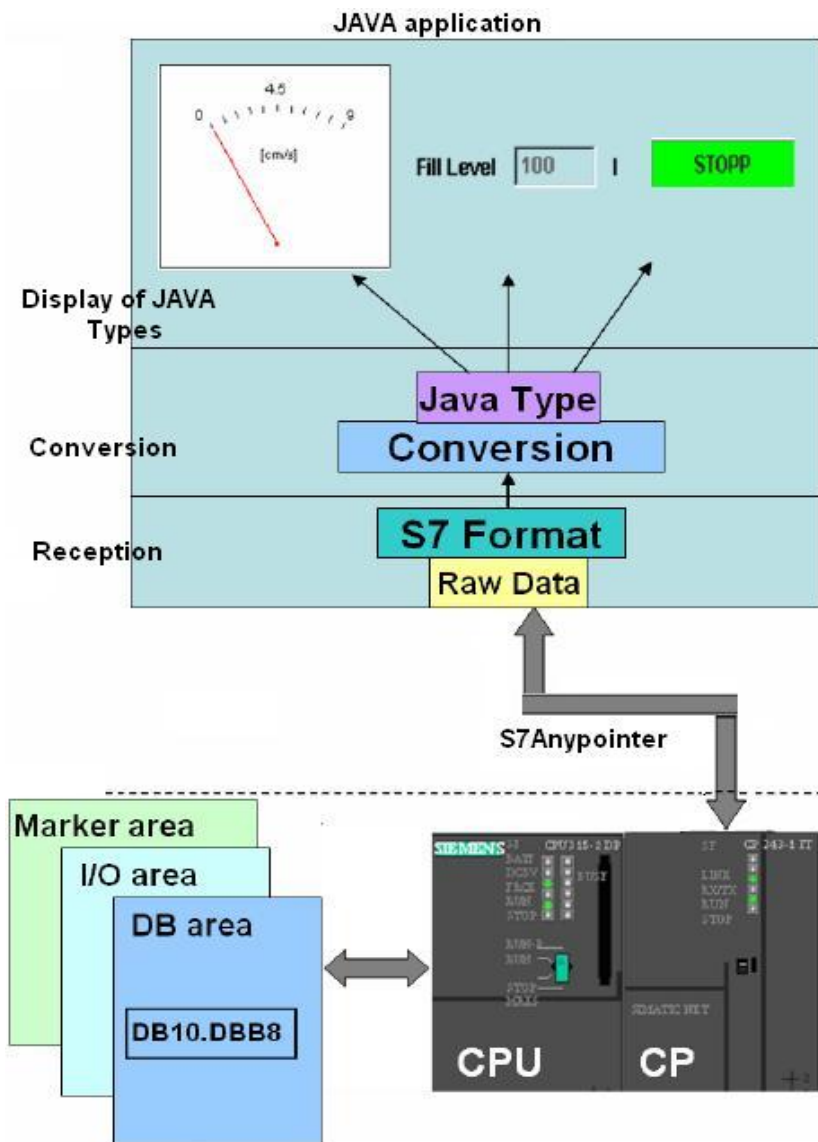


Figure 11.4 The structure of task

11.2.1 S7-Beans

Java beans are Java components that have built-in functions, and they can be controlled with known methods. Normally SIMATIC ITCP contains the some S7 beans, which manage the communication with the CP or CPU. These APIs can be downloaded and saved in ITCP manually as well. S7-Beans are saved in “Jar” archive files under three categories.

The S7 beans can be imported into the palette of components of a Java development environment like, for example Eclipse where they can be used and configured by the programmer using Drag-and-Drop in to visual class.

The following table shows the components after the import in a development environment.




Library	IDE Components	Significance
S7api.jar		device classes
S7gui.jar		visual components
S7util.jar		auxiliary classes for converting data types

Figure 11.5 API library V2.5.5

11.2.2 Hierarchical relationship between components

There is a hierarchical communicational relationship between S7CP, S7Device, and S7Variable, which is explained in Figure 11.6. This relationship is a copy of the relationship between hardware components. The program and variables are connected to the CPU; the CPU is connected to CP 343-1 IT. It is even easier to see in VisualAge program, where all the gadgets are connected through a linker (a line) with exactly same structure. Later we will see how this hierarchical relationship programmed in Eclipse.

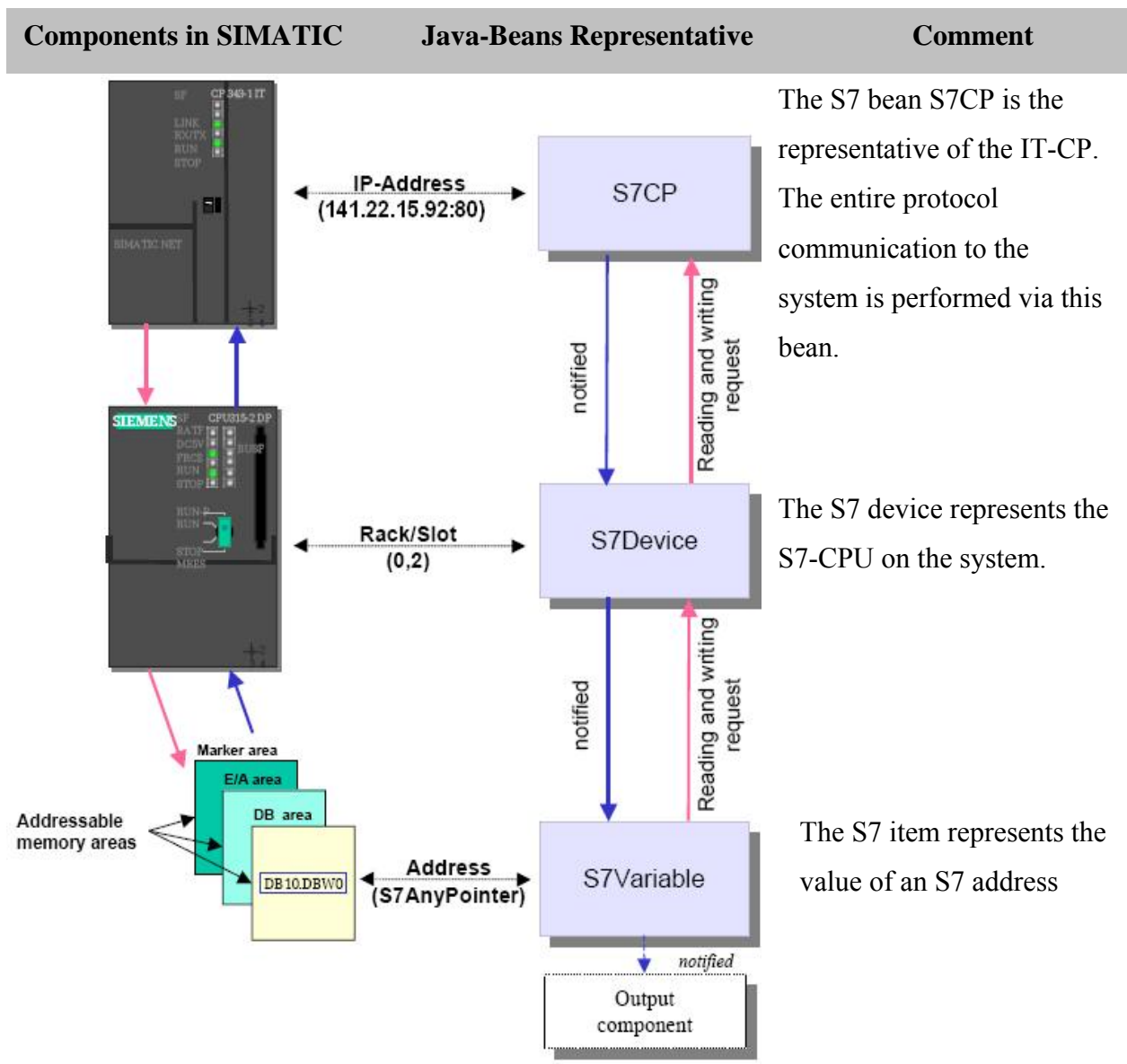


Figure 11.6 Comparison between HW and S7-Beans

When any of these object are modified, each object informs its objects connected to it via a Java standardized notification mechanism of the individual objects' PropertyChange methods.

Here is a piece of code, which realizes the structural relationship that is described in Figure 11.6. This code is written in Java, and it enables the communication between components.

```
// Declaration of the required components ----- 1

private S7CP s7CP1 = null;           // CP (Communication Processor)
private S7Device s7Device1 = null; // SIMATIC CPU
private S7Variable s7Variable1 = null; // Variable in the CPU
private CLTextOut cLTextOut1 = null; // a text box to connect Variable
private CLTimer cLTimer1 = null    // timer

public void propertyChange(PropertyChangeEvent evt) {
    // Query whether event was triggered by S7CP.
    if (evt.getSource() == s7CP1)
        // If YES
    // Pass event to the S7Device instance
    s7Device1.propertyChange(evt);
    // Query whether or not event was triggered by S7Device.
    // Connecting S7Device to CP ----- 2
    if (evt.getSource() == s7Device1)
        // If YES
    // Pass event to the S7Variable instance
    s7Variable1.propertyChange(evt);
    // Query whether or not event was triggered by S7Variable.
    // Connecting S7Variable to S7Device ----- 3
    if (evt.getSource() == s7Variable1)
        // If YES
    // Then transfer output value to the CLTextOut instance.
    //Connecting cLTextOut-bean to S7Variable ----- 4
    cLTextOut1.propertyChange(evt);
}
```

1. Declaration of the all the required components including textboxes, variables, and pointers.
2. The hierarchical communication start here from top to bottom; the communication processor CP 343-1 IT is connected to the SIMATIC S7 300 CPU
3. The variable is connected to the SIMATIC S7 300 CPU
4. The textbox is connected to the variable

11.3 Java Applet for displaying the HMI on a web browser

An applet is a software component that runs in the context of another program. Applets are small programs usually perform a particular piece of the overall user interface in a web page. There are some programs that are written in scripting languages, for example JavaScript, runs in the context of a larger program, but not considered applets.

Common examples of applets are Java applets and Flash movies, and Windows Media Player applet that are used to display embedded video files in Internet Explorer. Lately there are some plugins designed to display various 3D model formats in a web browser, via an applet. Many browser games are also applet-based, which come with various HMI that enable users to interact.

Web browsers, which are often equipped with Java virtual machines, can interpret applets from Web servers. Due to small file size, and cross-platformance compatiblity, and high security of applets (can't be used to access users' hard drives), they are ideal for small Internet applications that are accessible from a browser.

In this project the HMI is written in Java programming language, and it is displayed via applet.



Figure 11.7 Java virtual machine [7]

The below code is an example for an applet, which is displaying the first page of the HMI in the Web browser.

```
<html>
<head>
<title>IT_Automation</title>
</head>
<body>
<H1></H1>
<APPLET CODE = Page1.class    ARCHIVE = IT_Automation.jar
WIDTH = 1260 HEIGHT = 1024></APPLET>
</body>
</html>
```

Table 11.1 Java applet

<title>**IT Automation**</title> : This line writes the title of the Browser that opens to view the applet

APPLET CODE = **Page1.class** : This is the name of the Java Class that needs to be run in the applet.

ARCHIVE = **IT Automation.jar** : The archive file where the **Page1.class** file and other classes are archived as jar file.+

WIDTH = 1260 HEIGHT = 1024></APPLET : With and height lines define with and height of the visual object that displays the HMI in the Web browser.

11.4 Developing HMI in Java and connecting it with S7 Program

Up to now the control program was developed in S7-program, and its functionality was tested through diagnostic tools. The next step is how to visualize and run it through a Web page. HMI must be designed such a way that the end user can easily relate it to functionality of the system. The look and feel of HMI is limited by the S7 Java beans in the palette. Here are some of the beans provided in Java S7-Beans library:

11.4.1 Siemens S7 Beans

11.4.1.1 Siemens S7 Device beans

Device S7 Beans are in the package of API included in a Jar file called: s7api.jar





Siemens S7 Device beans	
S7-Bean	Function
 s7CP	This bean represents the IT-CP serving as the host. Any other IT-CPs that exist must be addressed using S7Device. This bean must be used with each applet for addressing and for saving the host address.
 s7Device	S7Device represents any intelligent S7 module such as a CPU, PROFIBUS CP, Ethernet CP, other IT-CPs (however, under no circumstances the IT-CP serving as host for the applets, to be addressed using the browser!)
 s7Variable	This bean represents variables in the S7 CPU.
 CLTimer	CLTimer is required for cyclic calling of methods of other beans. Whenever you want to monitor the status of an S7 module or a process variable continuously (cyclically), you require this bean. Note: CLTimer has no graphic representation

Table 11.2 S7-Beans for Devices

11.4.1.2 Siemens S7 GUI beans

GUI beans are included in a Java file called: s7gui.jar



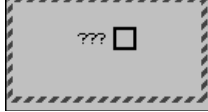


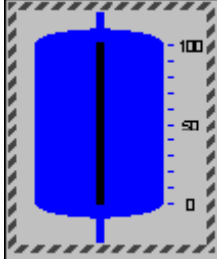

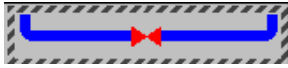
Siemens S7 GUI beans		
S7-Bean	Function	Representation
CLTexIn	CLTexIn is a bean for entering text. This text can be passed on to the S7 variable bean.	Input field
CLTextOut	CLTextOut is a bean for the textual output of values of process variables of the S7 variable bean	
CLIdentOut	CLIdentOut is a bean required for the textual display of an identification number of an IT-CP or module using the S7 CP bean or s7 device bean.	
CLStateLED	CLStateLED is a bean for graphic representation of the status of an IT-CP or a module.	
CLState3LED	CLState3LED is a bean for graphic representation of the status of an IT-CP or a module.	
CLTacho	CLTacho is a bean for graphic representation of a pointer instrument.	
CLLevel	CLLevel is a bean for graphic representation of the level of a process variable.	
CLPipe	CLPipe is a bean for graphic display of a horizontal or vertical pipe.	
CLValve	CLValve is a bean for graphic representation of a valve. The valve with its inlet can be displayed horizontally or vertically.	

Table 11.3 S7-Beans for GUI

11.4.1.3 Siemens S7 Utility beans

Utility beans are packaged in s7util.jar file

Siemens S7 Utility beans	
S7 Utility Bean	Function
DATE	Supplies the S7 type DATE as a string in the format D#2000-12-31.
TIME	Supplies the S7 type TIME as a string in the format T#9h6m6s.
DATEandTIME	Supplies the S7 type DATE_AND_TIME as a string in the format DT#00-12-31-12:31:47.487.
TIMEofDAY	Supplies the S7 type Time Of Day as a string in the format TOD#9:6:6.127.
S5TIME	Supplies the S7 type S5TIME as a string in the format S5T#1h3m2s.
ConvertNumberSystem	Supplies a decimal number in hexadecimal, octal, or binary format (as a string).

Table 11.4 S7 utility beans

11.4.2 HMI Design page 1

In chapter 11.1.4 configuration of a Java project with S7-Beans is described. Now we can continue using that project to design the first page to add a visual class so that we can design a GUI;

- file →new →Visual Class

The visual class in Eclipse editor's window will appear as it is seen in the Figure 11.8, which includes a 300 x 200 content pane.

11.4.2.1 Modifications to the class

11.4.2.1.1 Change layout

In the properties window (bottom) the “layout” property should be changed to “null” so that one can place the GUI components anywhere in the content pane. There is a palette on the right for entering GUI elements by mouse. The “Choose Bean” button in the palette enables one to choose beans to be used in the project.

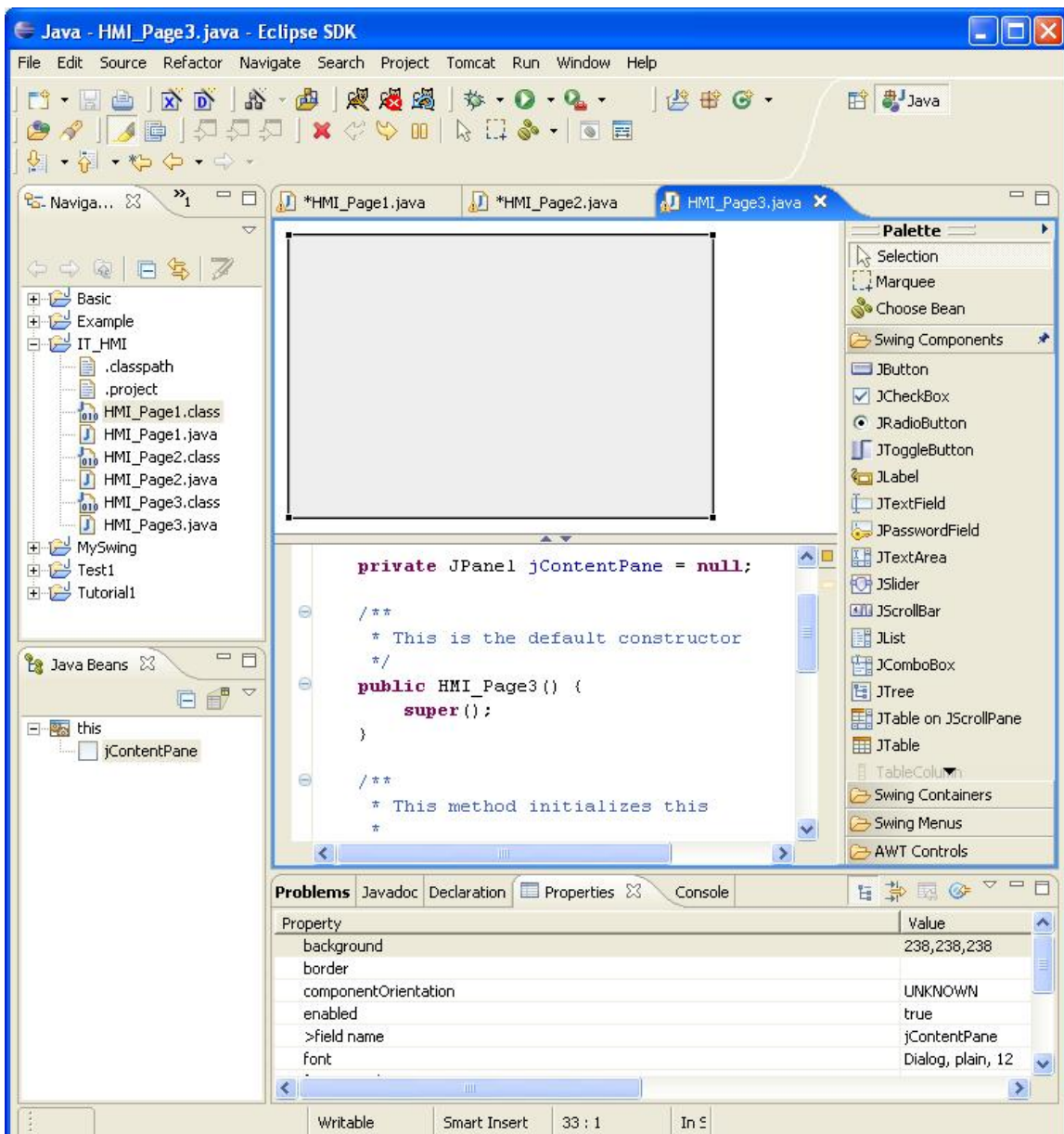


Figure 11.8 Visual project in Eclipse

11.4.2.1.2 *Modify the declaration line*

In the code window modify the line where the class is declared;

```
public class HMI_Page1 extends JApplet{...
```

should be modified to;

```
public class HMI_Page1 extends JApplet implements ActionListener,  
MouseListener, PropertyChangeListener{...
```

so that property changes, mouse actions will be detected.

11.4.2.1.3 *Add terminating API mechanisms*

All the applets with the S7-Beans must have these four methods in order to avoid resource problems;

```
public void start() {  
super.start();  
}  
  
public void stop() {  
super.stop();  
}  
  
public void destroy() {  
super.destroy();  
  
S7Api.terminate();  
}
```

Since the S7BeansAPI uses static resources and threads internally for communication with the IT-CP, the **terminate()** method was created in the **S7Api** class to release these resources and to stop all threads. **terminate()** can normally be called after releasing all its own resources in the **destroy()** method of the applet. [4]

11.4.2.2 **Inserting S7 Device Beans**

S7 device beans can be added to the visual class in two ways;

- Typing directly in the code window
- Inserting them from the “Choose Bean” button from the palette

The blow code (Device Beans) is manually entered by typing.

```
s7CP1 = new S7CP();  
s7CP1.setHostString(new HostString ("141.22.15.92:80"));
```

```

s7Device_1 = new S7Device();
s7Device_1.setRack(0);
s7Device_1.setSlot(2);

s7Variable1 = new S7Variable();
s7Variable1.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)10, (int)16, (int)1));

s7Variable1.setVariableName("s7Variable1");

```

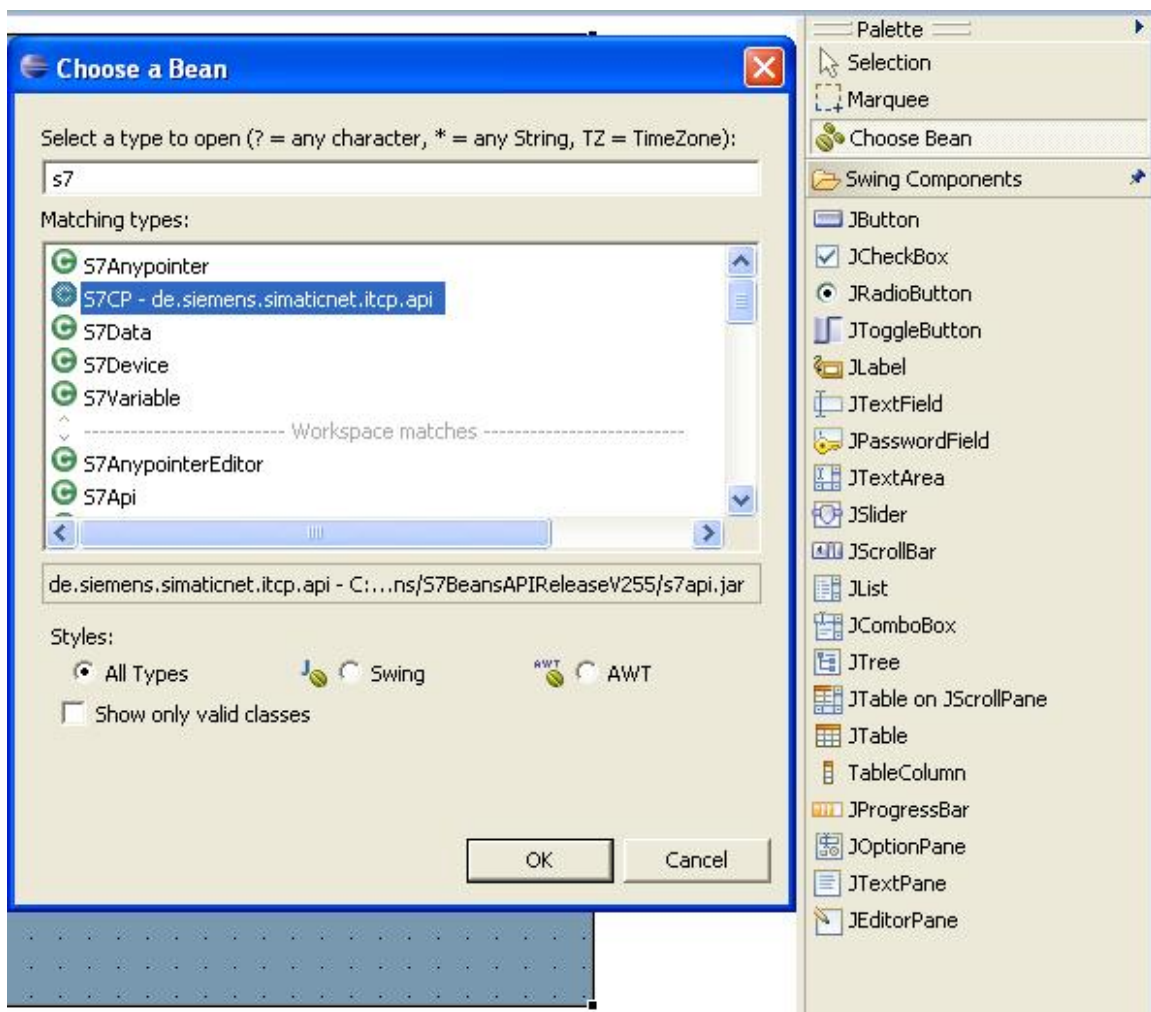


Figure 11.9 Adding beans from the palette

To be able to add S7 device beans from the palette one must remember or know first one or two characters of the name. Once the character entered the name area in the window showing above the list of beans will show up in the window, then it is easy to select one from the list.

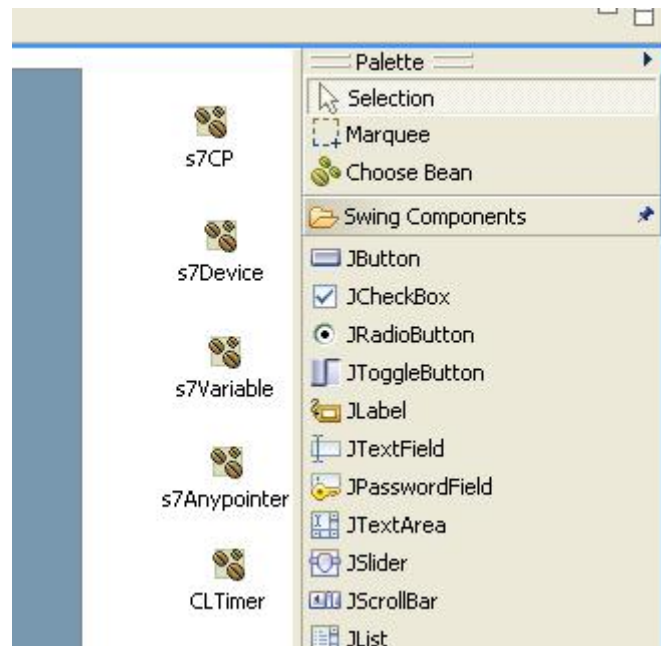


Figure 11.10 Device API inserted through palette

Here are some Device beans added to the project from the palette

- s7CP
- s7Device
- s7Variable
- s7Anypointer
- CLTimer

As it is described in this chapter (at 11.2) these beans must be connected hierarchically so that the information flow can be established from HMI to hardware and vice versa.

11.4.2.3 Design GUI

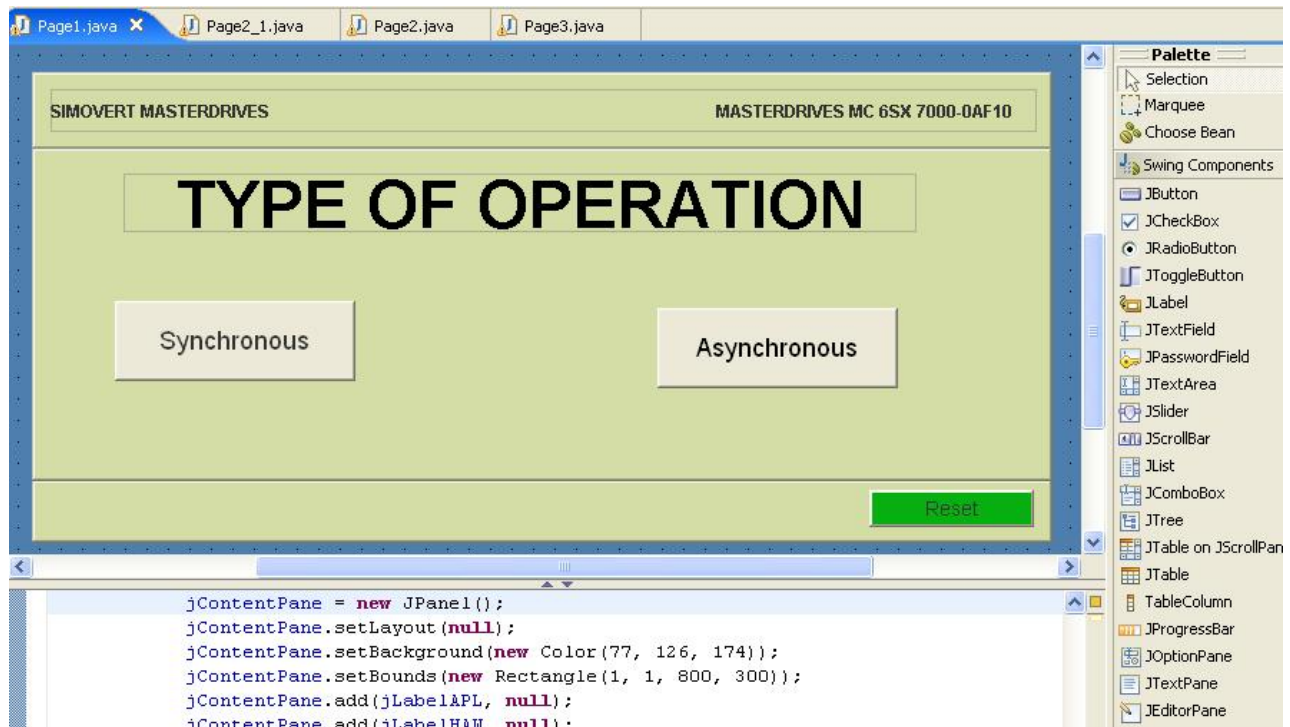
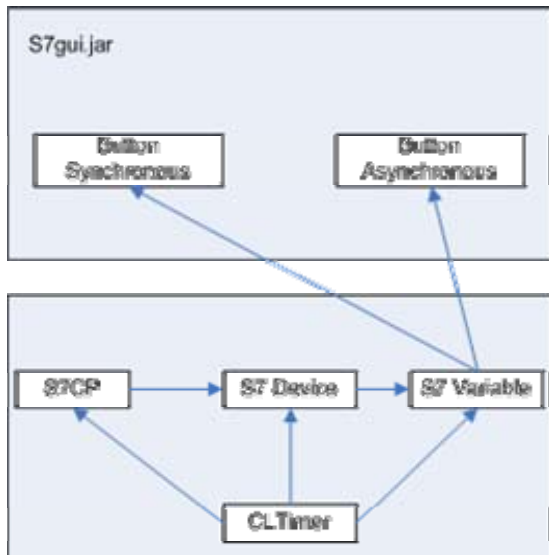


Figure 11.11 HMI page1

First the GUI is designed visually, which means the buttons and other GUI components added are not functional. To make them functional they have to be connected to S7 control program through an "S7Anypointer"s and "S7Variable"s.

The below code is generated by Eclipse when a button is added to the GUI.

```
private Button getButton_Page1() {
    if (Button_Page1 == null) {
        Button_Page1 = new Button();
        Button_Page1.setFont(new Font("Dialog", Font.PLAIN, 18));
        Button_Page1.setBounds(new Rectangle(58, 107, 168, 56));
        Button_Page1.setLabel("Synchronous");
    }
    return Button_Page1;
}
```



To add functionality to the button, which requires dealing with S7 program then a variable with an "s7Anypointer" must be declared and added to the class.

Figure 11.12 Components hierarch

The button "Synchronous" has two functions;

- Parameterize the Masterdrives MCs
- Enable the synchronous operation mode, when start button is pressed it should start the motors under this mode.

Here are the S7variables that perform these functions;

```
s7Variable1.setS7Anypointer(// to start Synchronous op.
new S7Anypointer((int)1, (int)1, (int)132, (int)10, (int)8, (int)1));

s7Variable4.setS7Anypointer(// to parameterize
new S7Anypointer((int)1, (int)1, (int)132, (int)10, (int)8, (int)3));
/*-----Bit number 0 ..7*/
/*-----Memory area offset*/
/*-----DB number or '0'*/
/*-----Memory area 132 == DB*/
/*-----Repetition factor 1 .. n*/
/*-----Data type 1 == BOOL*/
```

To understand the parameter values of S7Anypointer one can see the table 11.5. Let's take one of above S7Anypointers and explain the parameters according to the table 11.5.

```
S7Anypointer ((int)1, (int)1, (int)132, (int)10, (int)8, (int)3)
```

S7Anypointer	((int)1	(int)1	(int)132	(int)10	(int)8	(int)3
	Arg1	arg2	arg3	arg4	arg5	arg6

Argument 1: 1 Boolean

Argument 2: 1 repetition factor 1 means do the reading once

Argument 3: 132 Storage in a Data Block

Argument 4: 10 is the data block number (e.g. DB10)

Argument 5: 8 the start address of the memory are (only the byte section)

Argument 6: 3 the bit section of the memory area

So this S7Anypointer pointing the bit of 8.3 in the DB10. The toggle button (button “Synchronous”) is reading and writing this bit through the above S7Anypointer

Parameters	Description		
dataType	Data type of the variable (arguments) to be addressed		
	Data Type	Value (integer)	Significance
	BOOL	1	Bit
	BYTE	2	Bytes (8 bits)
	CHAR	3	Characters (8 bits)
	WORD	4	Words (16 bits)
	INT	5	Integers (16 bits)
	DWORD	6	Words (32 bits)
	DINT	7	Integers (32 bits)
	REAL	8	Floating point numbers (32 bits)
repFactor	Number of variable to be read (repetition factor)		
memArea	Area code to identify the memory area		
	Memory area	Value (integer)	Significance
	E	129	Input memory area
	A	130	Output memory area
	M	131	Flag memory are
	V	132	Storage are in Data Block
	SM	5	Special memory area
subArea	To specify DB number		
byteAddress	The beginning of the memory area for (byte section)		
bitAddress	The memory address (bit section)		

Table 11.5 parameter values of the s7Anypointer

The connection itself is not enough to define the functionality; there must be definitions for the button, which tells it how to act. For example the button “Synchronous” is a toggle button, when it is pressed it sends a “0” or “1” depending of the value of the specified bit.

First it checks the value, and then it writes a value to the destination address.

To check the value;

```
if (arg0.getSource() == s7Variable1){
    if(((Boolean)arg0.getNewValue()).booleanValue()){
        Button_Page1.setBackground(Color.green);
    }
    else{
        Button_Page1.setBackground(Color.red);
    }
}
```

To write the value;

```
if (arg0.getSource() == Button_Page1){
    if(((Boolean)s7Variable1.getValue()).booleanValue()){
        s7Variable1.setValue(String.valueOf(false));
        s7Variable1.waitOnNewData(100);}
    else{
        s7Variable1.setValue(String.valueOf(true));
        s7Variable1.waitOnNewData(100);}
}
```


11.4.3 HMI Design page 2 (Synchronous operation)

In Page2 also toggle buttons used for digital inputs (DI) for both sides. These buttons set the defined addresses “ON” and “OFF” depending on the status of the bit. The bit is read, and then the colour of the buttons is set. If the value of the bit is “1” the colour is set to green, if the value is “0” then it is set to red.

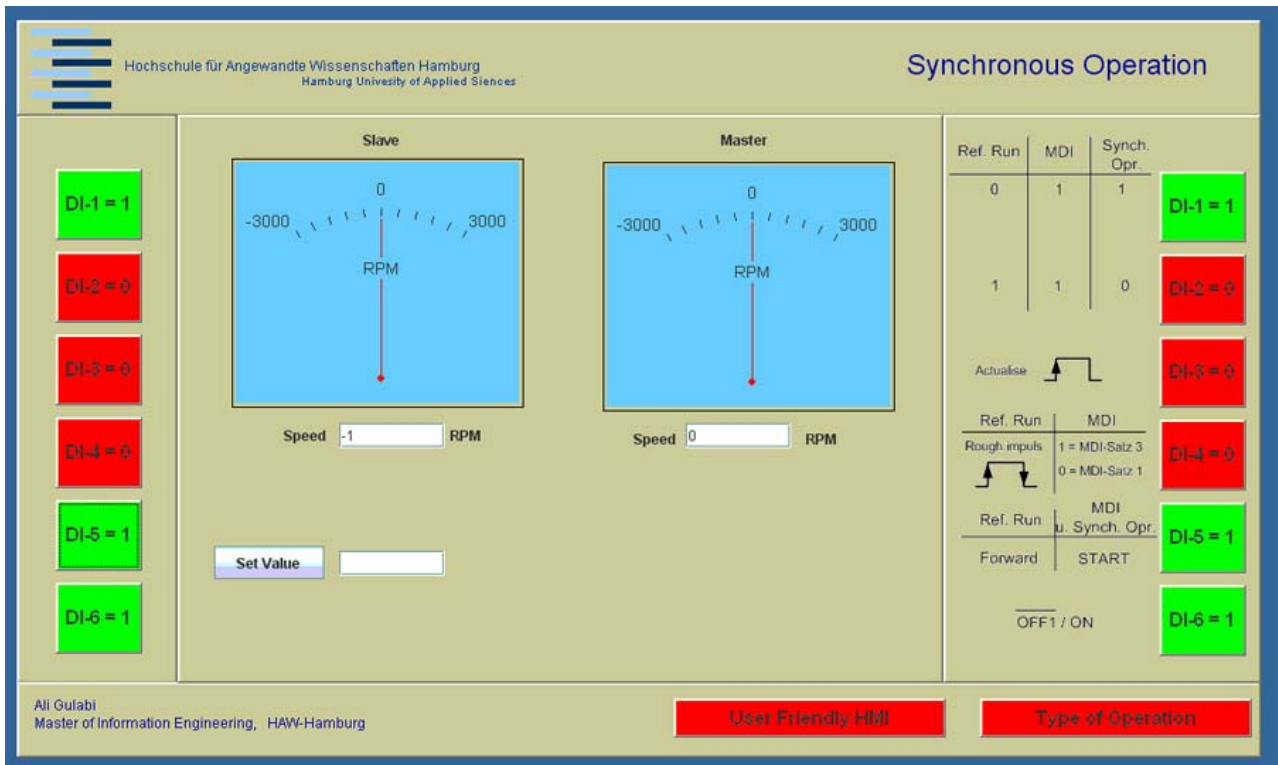


Figure 11.13 A screen shot of page2 of GUI

in previous section the connection of the toggle buttons are described, in this part therefore only connection of the tachos and text fields are going to be explained.

11.4.3.1 Adding Tacho

In eclipse editor do the following steps:

- Press the “Choose Bean” button in the palette
- When the window shows up; write CLTocho. You will see it in the list, and then double click on it.
- Now you are ready to add tocho any place on the pane.
- After adding it to the pane (GUI), click on the tocho then open the properties window in the bottom of the editor window.
- Define maximum and minimum speed, overflow value, background colour etc.

Connecting tacho to the S7Anypointer

To connect tacho to S7Anypointer a virtual wire should be defined, and then this wire will be used to connect tacho to S7Variable, which will be using S7Anypointer.

Virtual wire:

```
private void connEtoM31(java.beans.PropertyChangeEvent arg1) {
    try {getCLTacho_1().propertyChange(arg1);}
    catch (java.lang.Throwable ivjExc) {
    }
}
```

Connect the virtual wire to S7Variable (e.g.: s7Variable5)

```
if (arg0.getSource() == s7Variable5){
    connEtoM31(arg0);
}
```

The rest of the process for getting the value is the same as other variables, except the argument of the S7Anypointer must be chosen accordingly. Here is the CLTacho_1 S7Variable:

```
s7Variable5 = new S7Variable();
s7Variable5.setS7Anypointer(
new S7Anypointer((int)5, (int)1, (int)132, (int)100, (int)6,
(int)0));
s7Variable5.setVariableName("s7Variable5");
```

- **Arg1:** 5 INT (16 bit)
- **Arg2:** 1 repeat factor
- **Arg3:** 132 Storage in a data block (DB)
- **Arg4:** 100 points to DB100
- **Arg5:** 6 Start address byte section
- **Arg6:** 0 Start address bit section

So S7Variable5 reads speed as 16 bit integer value from the database 100 (DB100) at start address of 6.00.

11.4.3.2 Adding text box for displaying the speed

Enter a text field by pressing JTextField in palette of Eclipse. Change the properties of the field by using properties window. Here is the code generated by visual editor of Eclipse:

```
private JTextField getjTextField_speed() {
    if (jTextField_speed == null) {
        jTextField_speed = new JTextField();
        jTextField_speed.setBounds(new Rectangle(430, 261, 89, 21));
    }
    return jTextField_speed;
}
```

It has to receive the content (speed) from an S7Variable, so the connection to the S7 program can be established as shown below;

```
if(arg0.getSource() == s7Variable6){
    jTextField_speed.setText(arg0.getNewValue().toString());
}
. . .

s7Variable6 = new S7Variable();
s7Variable6.setS7Anypointer(
new S7Anypointer((int)5, (int)1, (int)132, (int)100, (int)6,
(int)0));
s7Variable6.setVariableName("s7Variable6");
```

This S7Variable reads an integer value from DB100, and then it is passed to the textfield to be displayed.

11.4.3.3 Adding text box for setting the speed

In order to set the speed; a value must read first, and then with an action it should be passed to a variable and an S7Snypointer, which saves the value to the memory location in DB100. For this reason a button is added to GUI, which is called “Set Value”.

Text field:

```
private JTextField getjTextField_set1() {
    if (jTextField_set1 == null) {
        jTextField_set1 = new JTextField();
        jTextField_set1.setBounds(new Rectangle(137, 369, 89, 21));
        jTextField_set1.setText("");
    }
    return jTextField_set1;
}
```

“Set Value” button:

```
private JButton getJButton_set() {
    if (jButton_set == null) {
        jButton_set = new JButton();
        jButton_set.setText("Set Value");
        jButton_set.setBounds(new Rectangle(31, 365, 93, 28));
    }
    return jButton_set;
}
```

Action: When the button is presses the value of text field will be passed to S7Variable;

```
if (e.getSource() == jButton_set) {
    s7Variable4.setValue(jTextField_set1.getText());
}
```

The variable is passing the speed value to the DB100 memory location 2.0 as an integer value.

```
s7Variable4.setS7Anypointer(
new S7Anypointer((int)5, (int)1, (int)132, (int)100, (int)2, (int)0));
```

11.4.3.4 Adding Digital Input (DI) buttons

The table below explains the connectivity of the control buttons with the memory areas via S7Variables. Technically they are also toggle buttons as explained in previous sections. They generate “0” or “1”, and the values are saved the specified locations that are listed in the table, and then these values transferred to the Masterdrives MC as control words by S7 control program.

Digital Input (DI)	Significance	Bit address in DB100		Control buttons
		Right side	Left side	
DI-1	Operation mode	10.0	13.0	DI 1
DI-2	Operation mode	10.1	13.1	DI 2
DI-3	Acknowledge	10.2	13.2	DI 3
DI-4	Reference operations	10.3	13.3	DI 4
DI-5	Start/Stop	10.4	13.4	DI 5
DI-6	ON/OFF	10.5	13.5	DI 6

Table 11.6 Control buttons and addresses

Right Side:

```
s7Variable1.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)10, (int)0));
s7Variable2.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)10, (int)1));
s7Variable3.setS7Anypointer(
```

```

new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)10, (int)2));
s7Variable9.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)10, (int)3));

s7Variable10.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)10, (int)4));
s7Variable11.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)10, (int)5));

```

Left Side:

```

s7Variable14.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)13, (int)0));
s7Variable15.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)13, (int)1));
s7Variable16.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)13, (int)2));
s7Variable17.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)13, (int)3));
s7Variable21.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)13, (int)4));
s7Variable19.setS7Anypointer(
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)13, (int)5));

```

11.4.3.5 Adding navigation buttons

After adding the button to the GUI, the button set to listen actions. When it is pressed, it will open the Page2_1.htm, which is the user friendly HMI.

```

if("User Friendly HMI".equals(e.getActionCommand())){
    AppletContext am = getAppletContext();
    try{
        am.showDocument(new URL(getCodeBase(), "Page2_1.htm"));
    }catch (MalformedURLException e1) {
        }
    }
}

```

11.4.4 User friendly HMI design Synchronous operation

Each drive has 6 buttons in the HMI for Synchronous operation (see figure 11.13) to operate under different modes (e.g.: synchronous operation, reference run, reference run for 28 rpm, and etc.). These operation modes and buttons are standard buttons for the SIMATIC MASTERDRIVERS MC training set (6SX-7000-0AF10 see figure 1.1). If one knows how to operate the training set then he or she can run the system from the synchronous operation HMI which designed in this project. The only problem is that the trainings set user interface is not very user friendly. Therefore, the reason for a user friendly design is to make it more self descriptive.

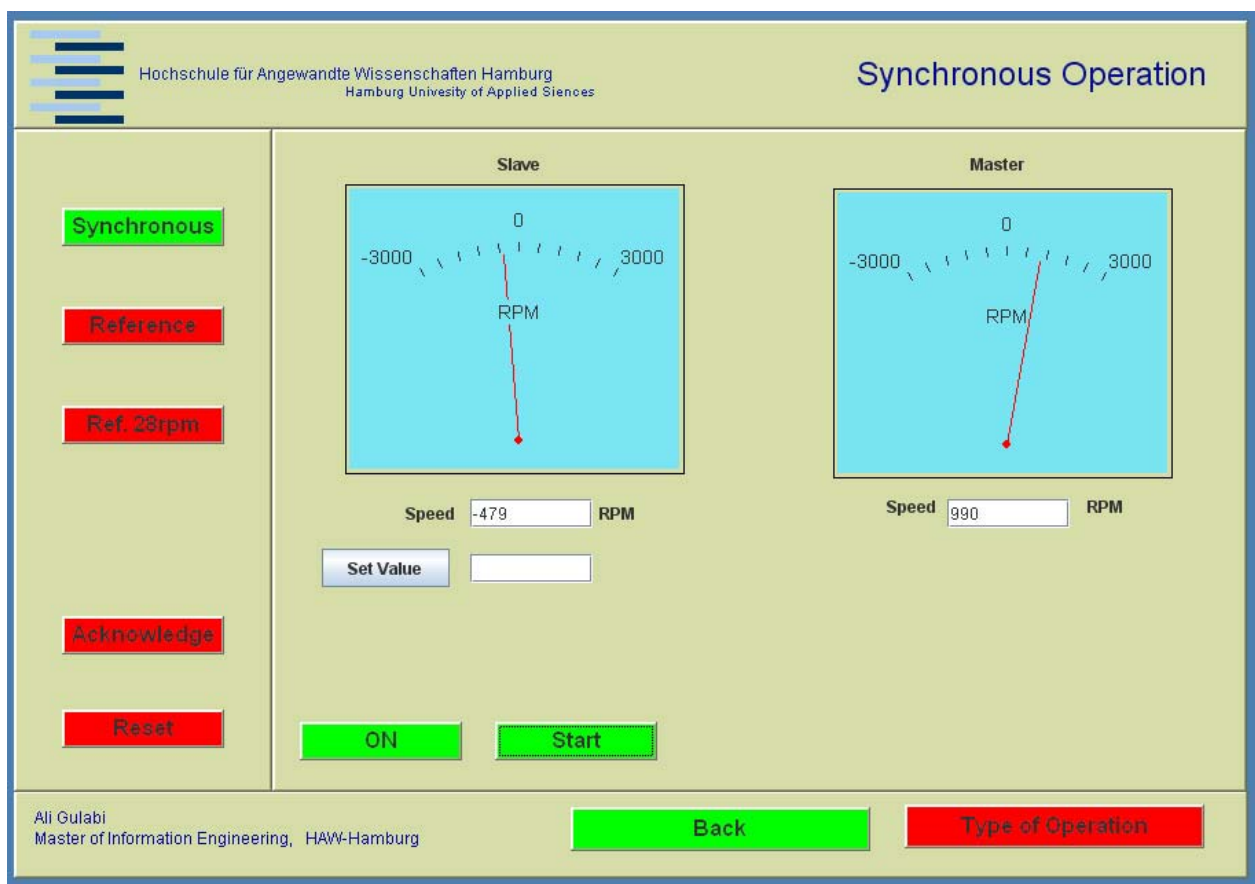


Figure 11.14 User friendly HMI

There are no difference between the Synchronous mode and user friendly Synchronous mode HMI in terms of the functionalities, and Masterdrives MCs parameterizations. The only difference is that same digital inputs (DI) for both drivers are directed (controlled) to a single button, which is labelled with the operation mode. Therefore the user does not need to know about the all combinations of the operation modes buttons on the training set. For example to

operate under synchronous mode; one only needs to pres “Synchronous” then “ON”, and then “Start” buttons. The “Synchronous” button is controlling DI-1s for the both drivers.

Here are the S7Variables that perform the control mechanism;

```
s7Variable1.setS7Anypointer(//-----Synch Op.-----  
new S7Anypointer((int)1, (int)1, (int)132, (int)60, (int)0, (int)0));  
s7Variable2.setS7Anypointer(//-----Reference Op.-----  
new S7Anypointer((int)1, (int)1, (int)132, (int)60, (int)0, (int)1));  
s7Variable3.setS7Anypointer(//-----Reference Op28rpm.-----  
new S7Anypointer((int)1, (int)1, (int)132, (int)60, (int)0, (int)2));  
s7Variable10.setS7Anypointer(//-----Acknowledge-----  
new S7Anypointer((int)1, (int)1, (int)132, (int)60, (int)0, (int)4));  
s7Variable11.setS7Anypointer(//----- Start -----  
new S7Anypointer((int)1, (int)1, (int)132, (int)60, (int)0, (int)5));  
s7Variable9.setS7Anypointer(//----- Back to Synch Op. page-----  
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)28, (int)0));
```


11.4.5 HMI Design for Asynchronous operation

The HMI for asynchronous operation is a reduced version of HMI of Synchronous operation. Just because all the program functionalities are implemented in the S7 control program the only difference between synchronous and asynchronous operations comes with the connectivity of the control buttons, where they are enabling/disabling relevant part of the S7 control program. Another difference is there are two speed setting buttons, one for each drive, where each drive is operated independently.

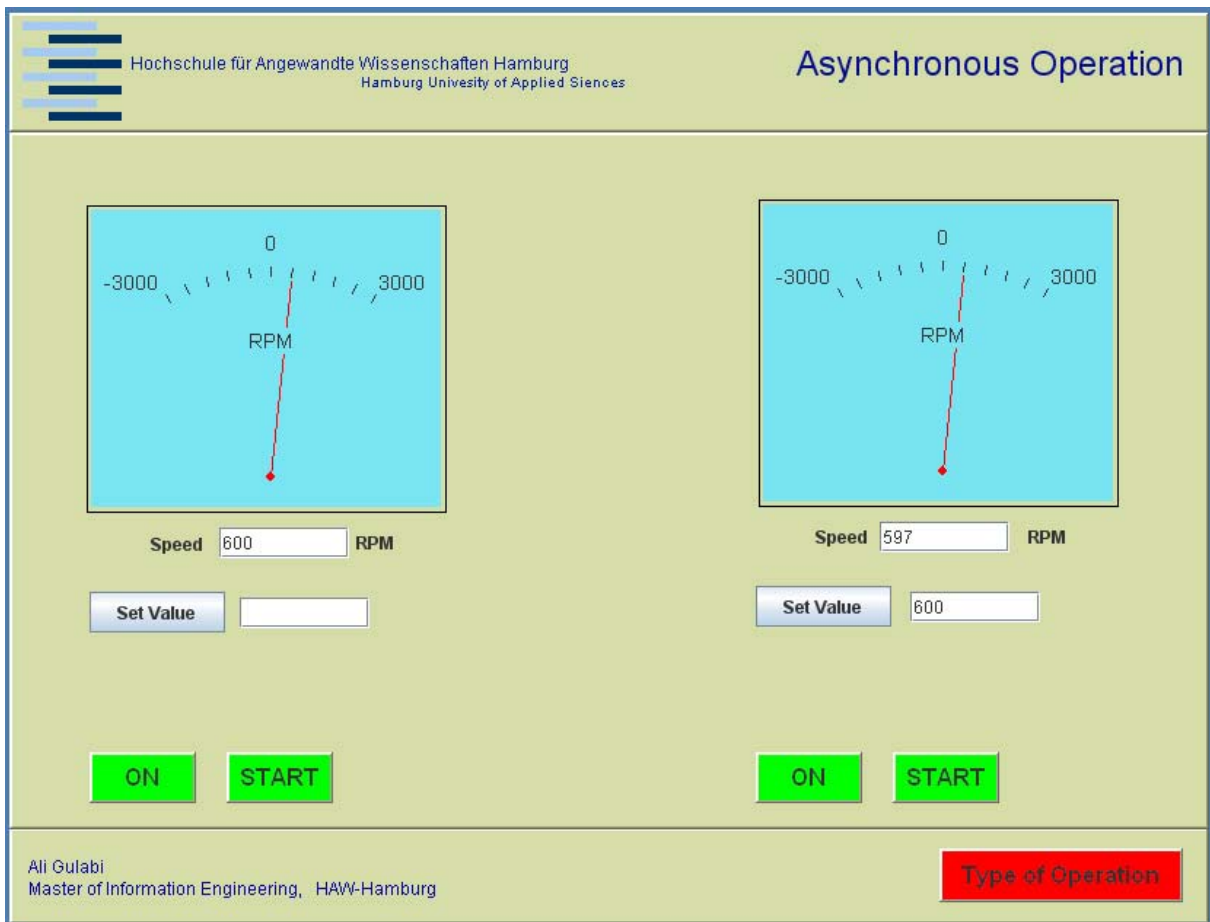


Figure 11.15 HMI Asynchronous Operation

```
s7Variable2.setS7Anypointer( //Start Right
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)10, (int)0));
s7Variable3.setS7Anypointer( //ON Right
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)10, (int)1));
s7Variable9.setS7Anypointer( //Start Left
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)13, (int)0));
s7Variable10.setS7Anypointer( //ON Left
```

```
new S7Anypointer((int)1, (int)1, (int)132, (int)100, (int)13, (int)1));
```

The “Type of operation” button has two tasks, as they are on the other HMIs, navigating to the main page, and resetting the parameters so that the new operation will be able to start.

12 Conclusion

The speed of growth in communications, computer science and semiconductor technologies is directly affecting many other industries including automation. Nowadays it is desired to have safer and more accessible automation systems. Today it is possible to control, and manage life threatening chambers without jeopardizing anybody's life with help of these new methods in automation.

The internet is one of the tools that find application areas in almost every industry. The term SCADA usually refers to a central system that monitors and controls a complete site or a system spread out over a long distance. It makes good use of the Internet to accomplish the connection. Due to development in quality of service internet services; it is possible to meet the requirements of the SCADA in automation systems.

The goal of this master thesis was to develop an Embedded SCADA system with PLC and Java Application for Synchronous Operation of Standard Servo Drives. The SCADA system has two tasks; Data Acquisition, and Supervisory control. In the network system the data flow is bidirectional therefore the both tasks can be performed at the same time.

In the system, Masterdrives MCs work standard PLC peripheral equipment to produce the process data. The CPU 315-2 DP is the master device of the system. It is responsible for data acquisition from the Motion Controllers via PROFIBUS-DP. On the other site, CPU 315-2 DP is directly connected with CP 343-1 IT and transfers the process data periodically via TCP/IP connection. The user in the client site can access and send the process data through the Internet using the web embedded HMI.

The Human Machine Interface of the system is developed with Java Applets, which is a cost effective and a platform-free method. Although in this project it seems to be doable for the automation systems, one must test it with more complex networks with heavy data load.

12.1 Suggestions for the future work

Even though this project meets the requirements, and ready to deploy it in real environment there has to be some improvements. They can be categorized in three groups; first two groups are software related, and the last one is a hardware related one.

- Improvement in Java programming
- Improvement in S7 control program
- Improvement in hardware

Test program modules can be used to find out the reason behind slow down in HMI, which is written in Java. There are few things can be investigated; number of the S7Variables, program structure, and time intervals of the timers.

If number of S7Variables is the problem then a Step7 program can be developed to take most of load from Java program. For example instead of separate DI and S7Variables one can combine the entire DIs in to one S7Variable, let S7 program do the work.

Second improvement can be done to improve error handling. Since Java programming in Eclipse is done manually (compare to VisulAge) it is important to test the program for every scenario and fix the bugs.

Third improvement can be done by using newer versions of hardware. For example instead of using SIMATIC S7-300 one can try S7-400. Masterdrives MCs also a bit old the firmware was updated from 1.3 to 1.66. They are not compatible with the newest firmware. Maybe Motion controllers with newest firmware enable us to control better.

13 References

- [1] Siemens document: PI_ATEX1_x.pdf
- [2] Siemens document: A01_TIA.pdf
- [3] https://www.automation.siemens.com/net/html_76/produkte/040_produkte.htm?HTTPS=RED
[IR](#) 06.03.07
- [4] Siemens document: mc166_kompend_e.pdf
- [5] http://en.wikipedia.org/wiki/User_interface 09.04.2007
- [6] <http://www.eclipse.org/> 10.04.2007
- [7] <http://www.java.com/en/download/manual.jsp>
- [8] Information Technology with the CP 3431 IT and CP 443-1 IT C79000 G8976-C120-0
- [9] S7-300, CPU 31xC and CPU 31x: Hardware and installation, Siemens AG,
GES73988FA10-8BA0
- [10] SIMATIC NET IT-CP Programming Tips, Siemens AG, C79000 G8976-C153-02
- [11] Industrial Ethernet, Siemens, <http://www.ad.siemens.de>
- [12] S7Beans/Applets Programming Tips, Siemens AG, C79000 G8976-C153-01
- [13] Instructions for the CP 343-1 IT and CP 443-1 IT C79000-G8976-C120 Release 04
- [14] Statement List (STL) for S7-300 and S7-400 Programming, Siemens AG, 1-4
6ES7810-4CA07-8BW1
- [15] Ladder Logic (LAD) for S7-300 and S7-400 Programming, Siemens AG,
6ES7810-4CA07-8BW1
- [16] Process Control System PCS 7 Programming Instructions for Blocks, Siemens AG,
Manuel
- [17] Programming with STEP 7 V5.3, Siemens AG, 6ES7810-4CA07-8BW0

Appendix

A Configuration

A.1 PC

In order to configure SIMATIC S7-300; first it has to be connected through MPI. MPI connection is ready for any kind of upload and downloads operations before system configuration is complete. In production floor MPI connection is widely used for engineering support to re install the system configurations. A PROFIBUS network system configuration can only be downloaded using MPI connection. Therefore we are going to configure MPI first, and when the PROFIBUS bus system is configured it will be downloaded to the SIMATIC S7-300 CPU, and then connection will be switched to PROFIBUS.

1. Configure PG/PC to MPI mode to be able to communicate with S7-300 CPU. For this open the PG/PC from program menu.

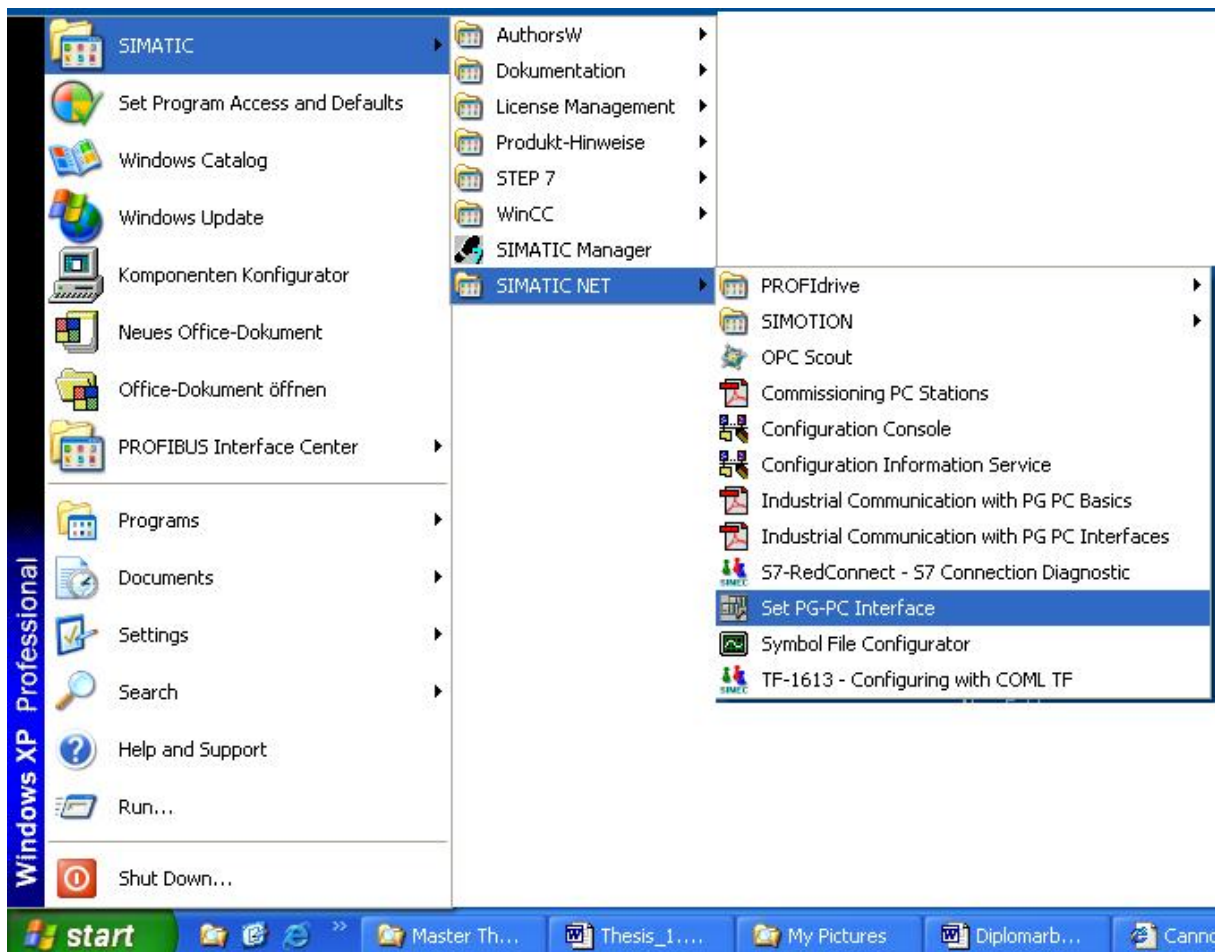


Figure A.1 Starting PG-PC Interface

2. Set CP5613A2 (MPI) active.

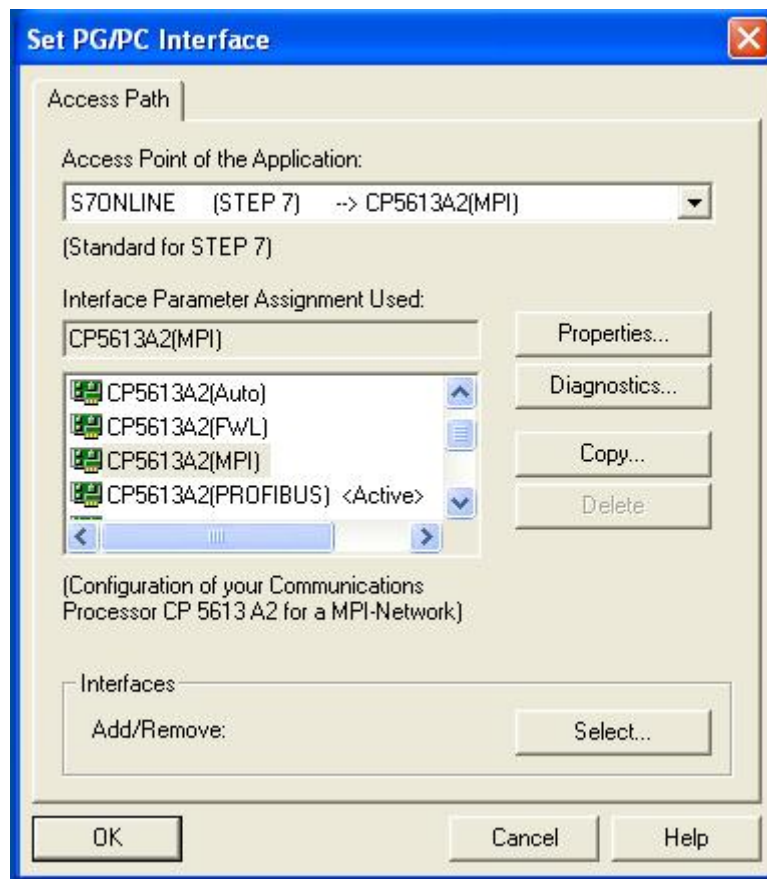


Figure A.2 Activate MPI

3. Connect the PG cable to the output of the CP5613A2 communication processor and MPI-Bus socket on CPU.

A.2 S7-Hardware Configuration

1. Create the S7-Project



2. Double click SIMATIC Manager icon to start S7.

3. To create a new project; File → New

When the dialog window opens name the project (e.g.: IT_Control).

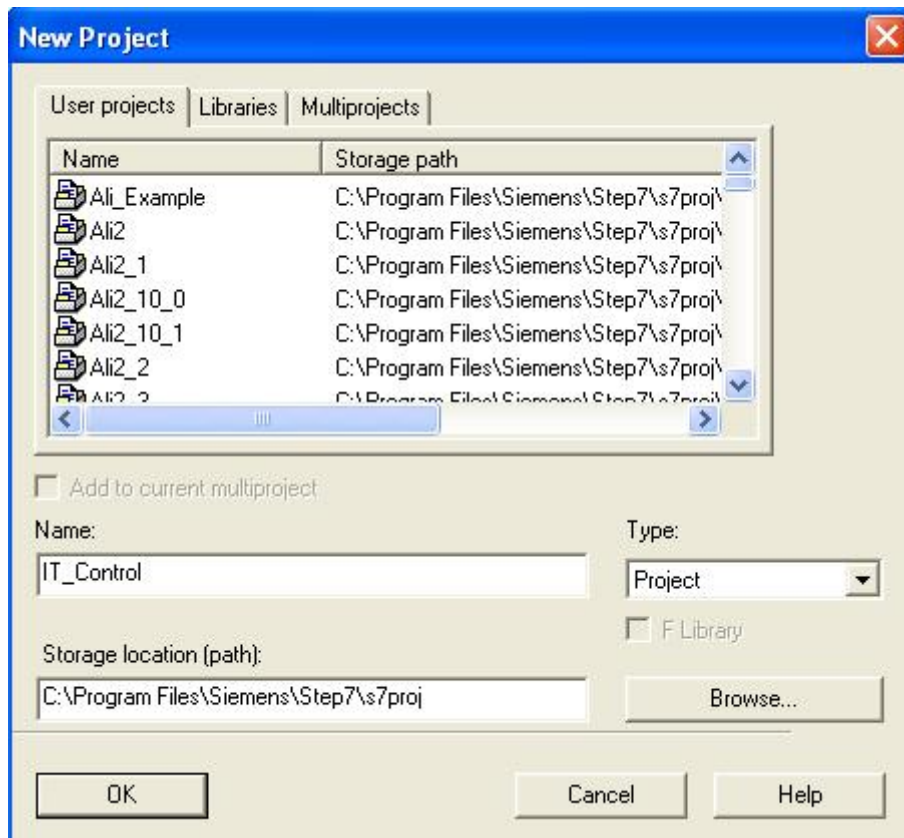


Figure A.3 Create a new project

4. As we can see from the Figure A.4 the project only includes the MPI bus, for other hardware components we must right click on the project icon, and enter the component that we need.

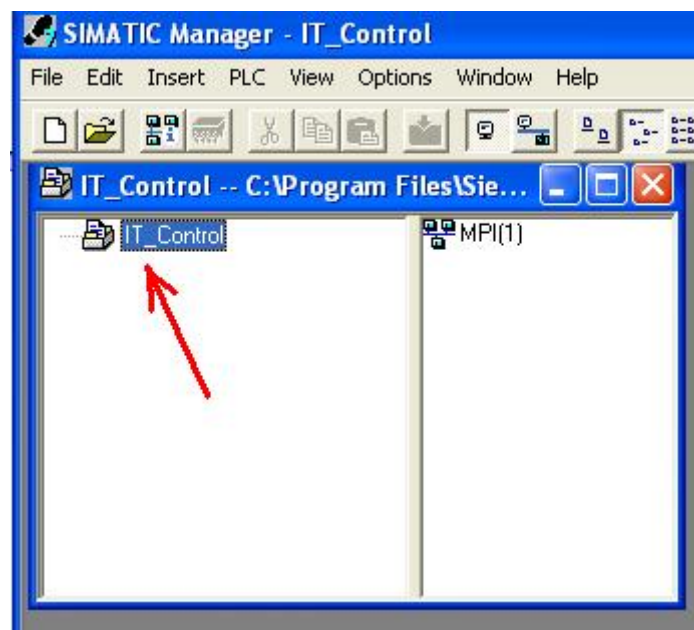


Figure A.4 Entering hardware components

- 5. Insert SIMATIC 300 station. To do this; right click on the “IT_Control” project →Insert New Object→ SIMATIC 300 Station.

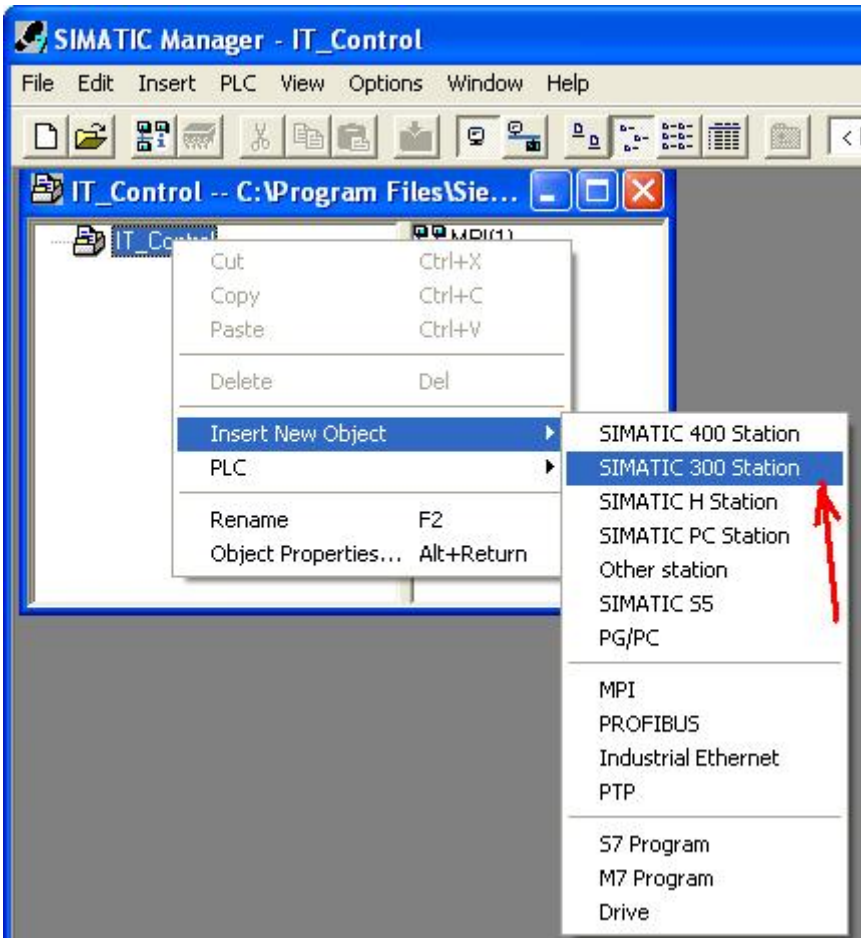


Figure A.5 Enter SIMATIC 300 Station

- 6. Insert a subnet “Industry Ethernet”

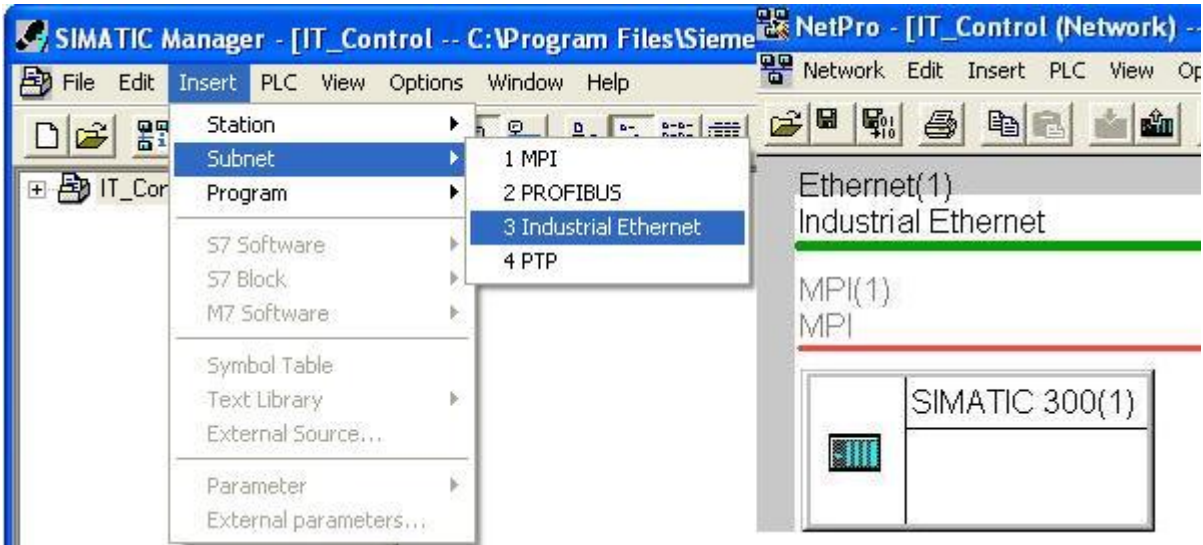


Figure A.6 Enter Industry Ethernet

A.2.1 Hardware configuration

- 1. Double click on hardware icon to open the hardware configuration view.

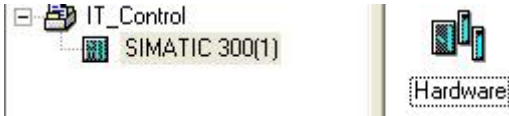


Figure A.7 Hardware configuration

- 2. Press the catalog button to open component list on the right.

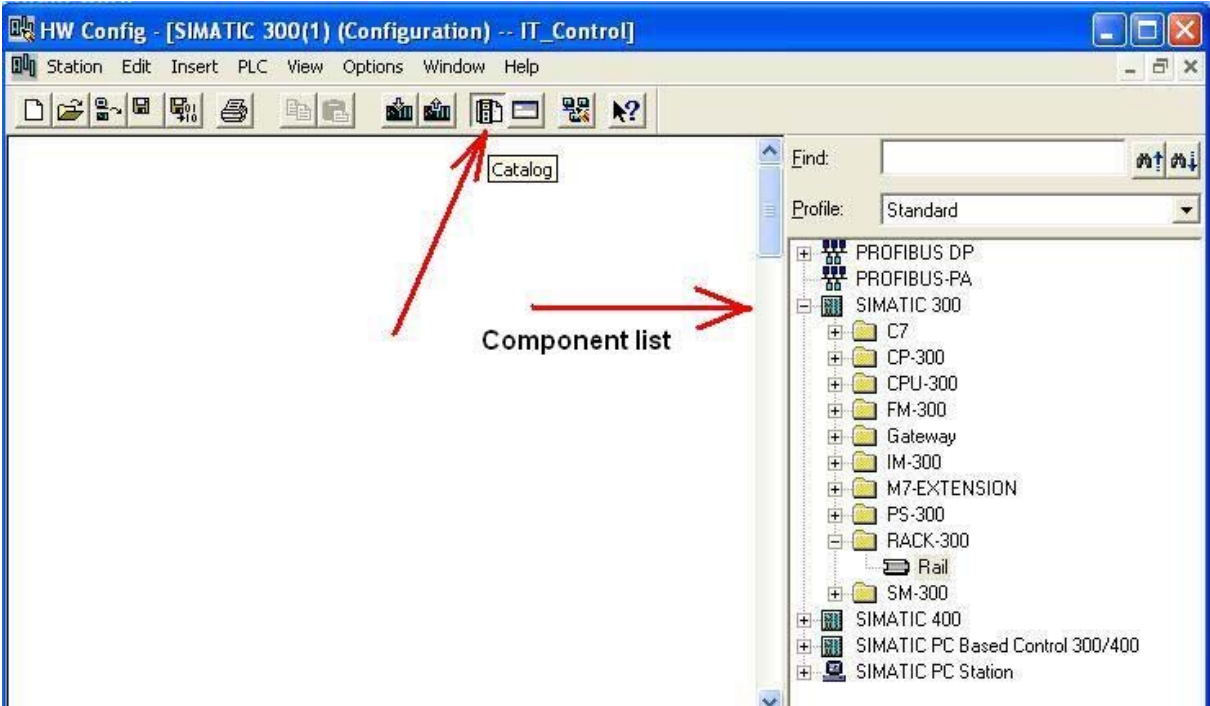


Figure A.8 Component catalogue

3. Insert the “rail” the rack which the components are going to be mounted on. From the catalog →SIMATIC 300→RACK-300→Rail

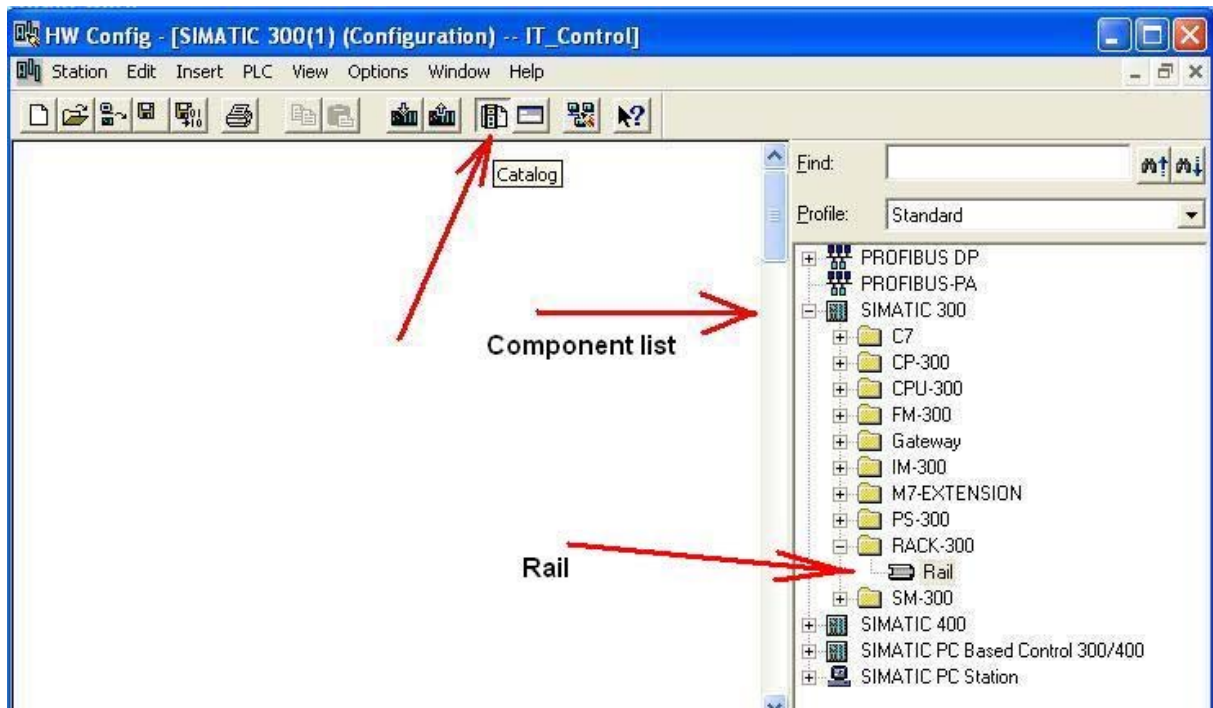


Figure A.9 Insert rail

4. Add a power supply: in the catalog →PS 300→PS 307 5A. PS 307 5A is the power supply that is mounted my physical set up, so every time we add a component in the project it must match with the product name and model in the physical hardware setup.

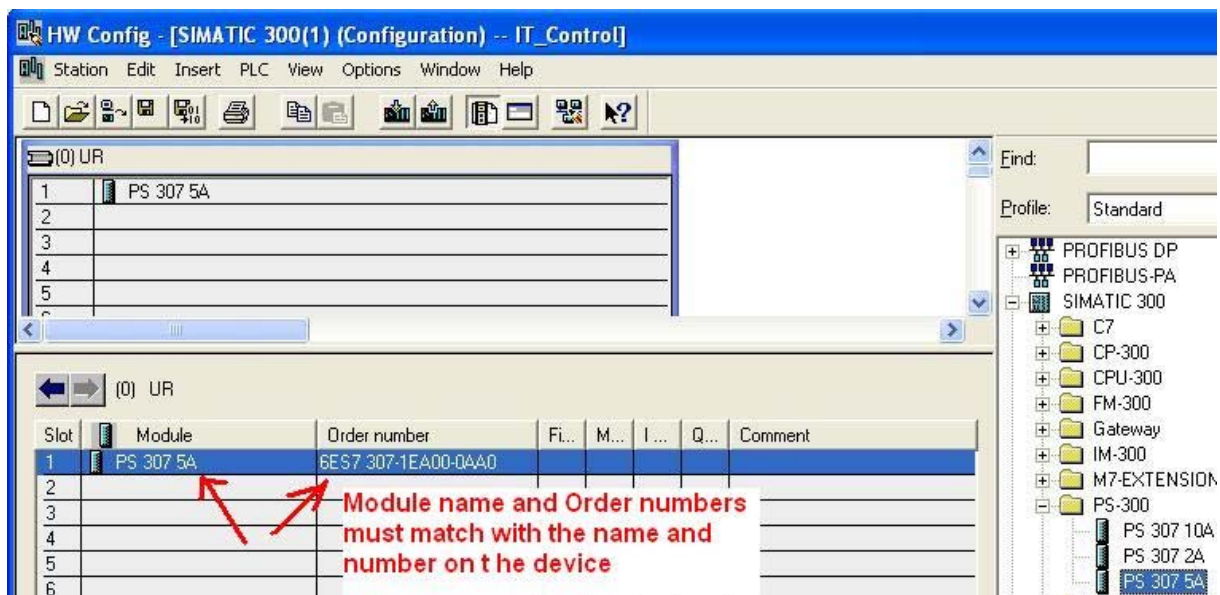


Figure A.10 Add power supply

5. Insert the CPU; “CPU 315 2 DP”

SIMATIC 300 → CPU 315-2 DP → 6ES7 315-2AF03-0AB0 → V1.2

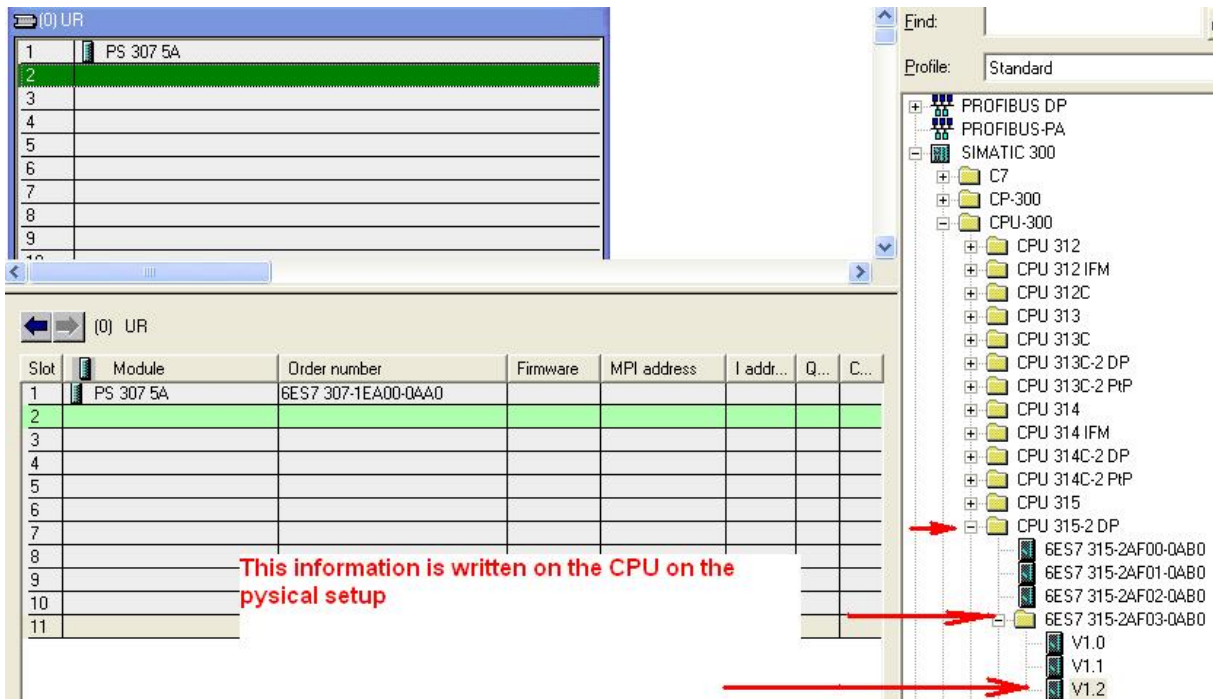


Figure A.11 Insert CPU

6. Set the properties of the CPU as MPI network and data rate to 187.5 Kbps. To do this; double click on CPU → General → Properties → New

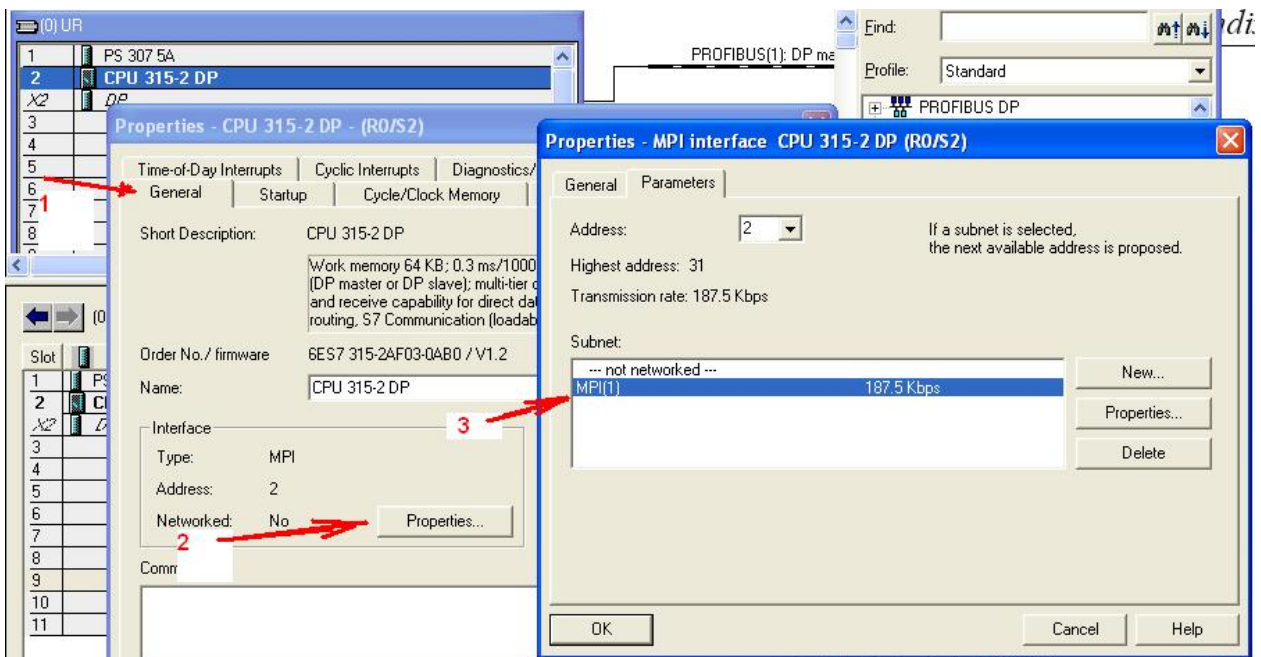


Figure A.12 MPI properties

7. Insert Communication Processor “CP 343-1 IT”: SIMATIC 300 → CP-300→ Industrial Ethernet → CP 343-1 IT → 6GK7 343-1GX20-0XE0

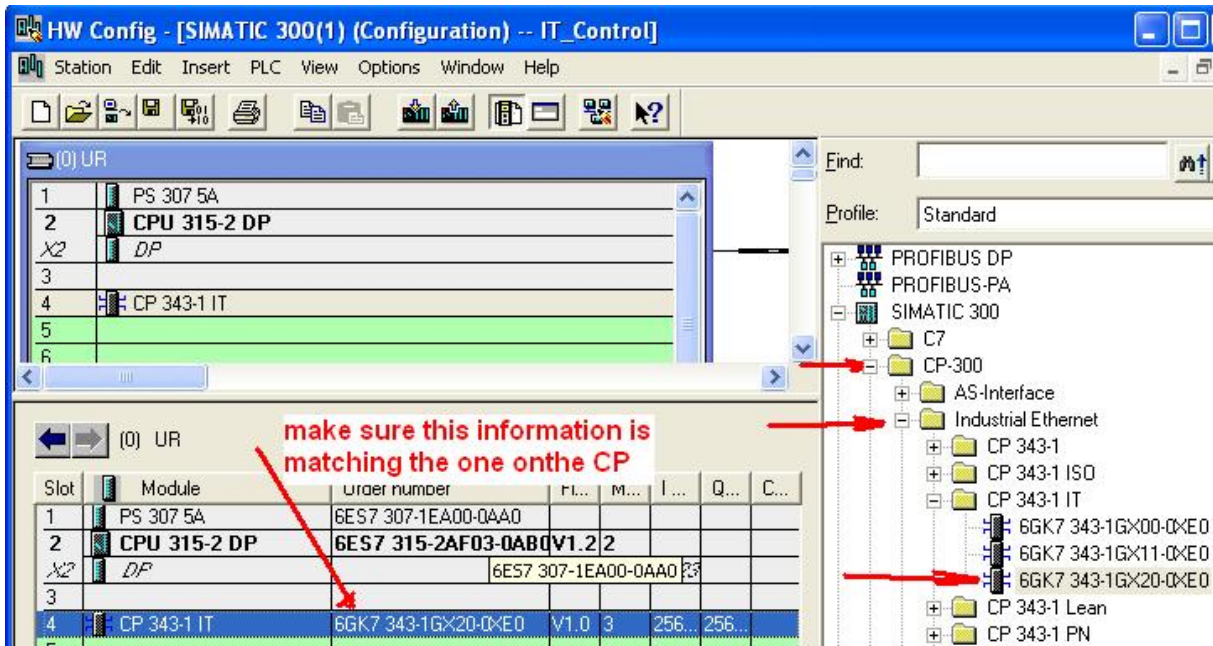


Figure A.13 Add communication processor

8. Set IP address of Communication Processor CP 343-1 IT. The IP address is written on the CPU. In this case it is 141.22.15.92. Double click on “CP 343-1 IT” → General → Properties → New

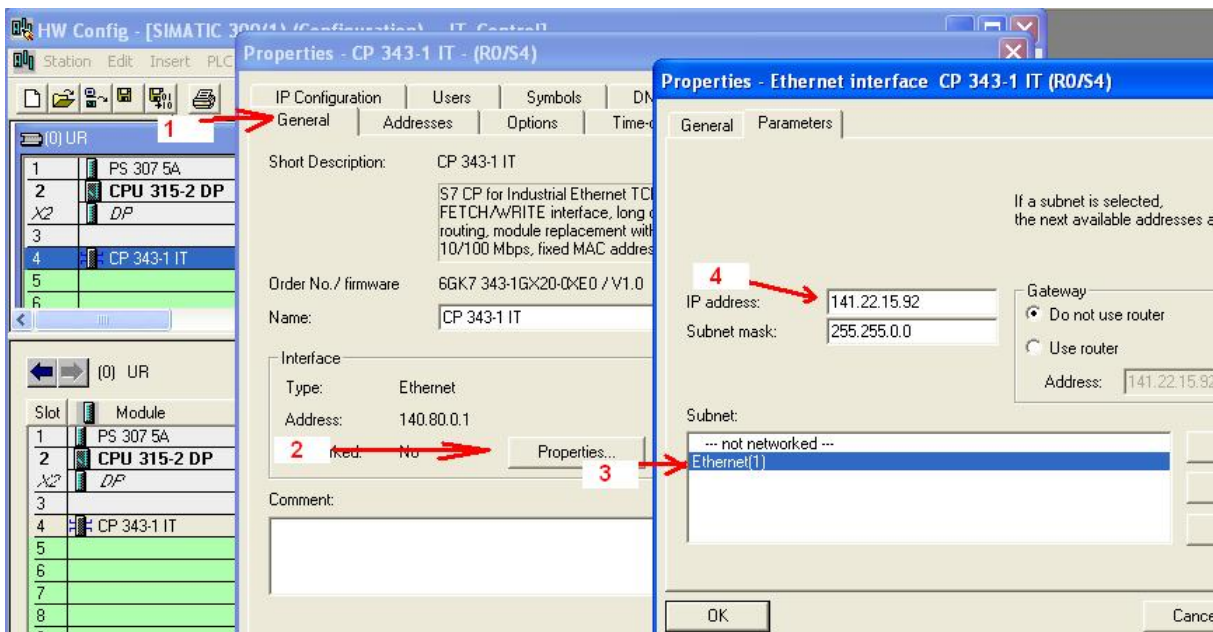


Figure A.14 Set IP address

9. Set Time-Of-Day Synchronization Select NTP Mode → Active NTP time-of-day synchronisation, Add NTP Server Address: 131.183.3.220, Select Time zone: (GMT + 1:00)Berlin
10. Add the users in the CP 343-1 IT Users properties window, and set the user rights and enter the password. Double click “CP 343-1 IT” →Users→Add

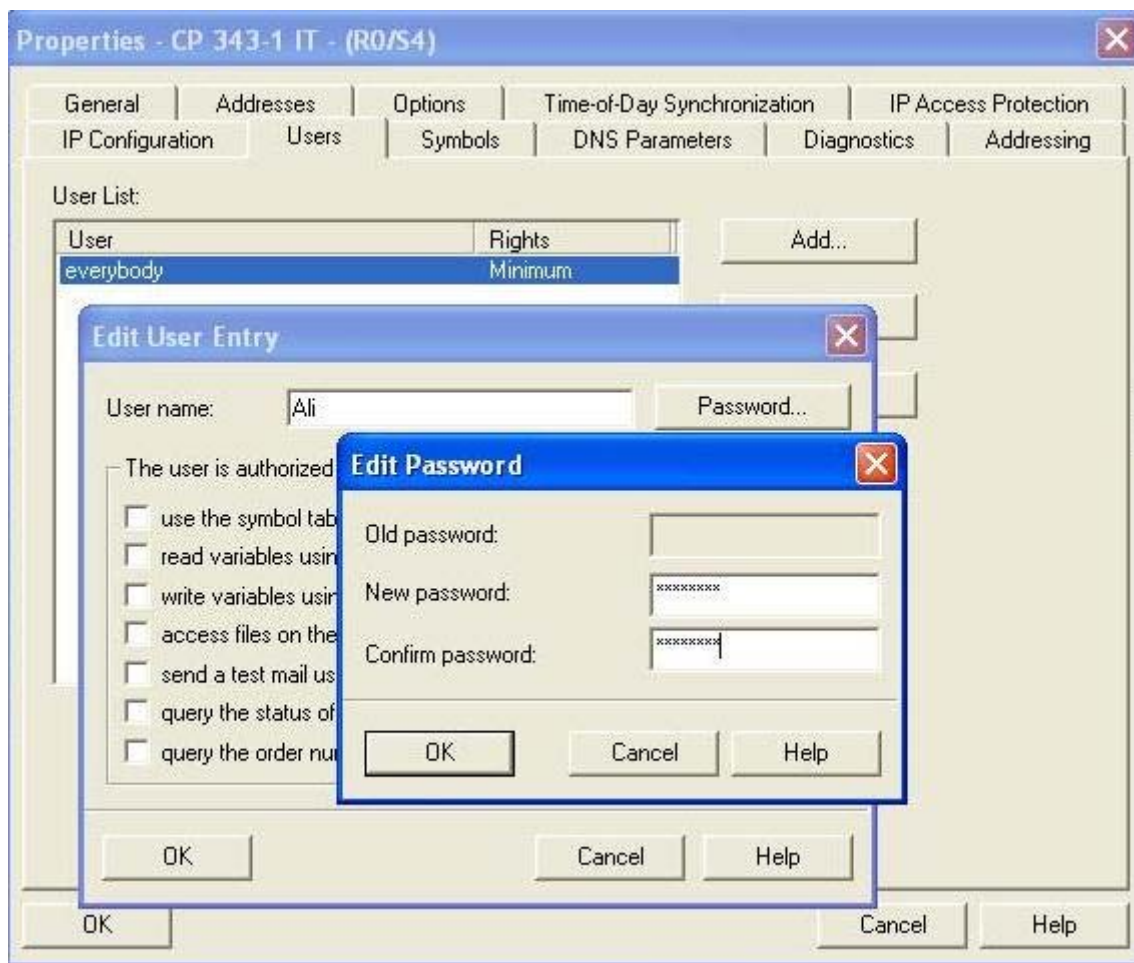


Figure A.15 Add users

11. Set DNS server address,. Double click “CP 343-1 IT” →DNS Parameters →Add
12. Insert Digital Input module SM 321-1BH02-0AA0
SIMATIC 300 → SM-300 → DI-300 → SM 321-1BH02-0AA0

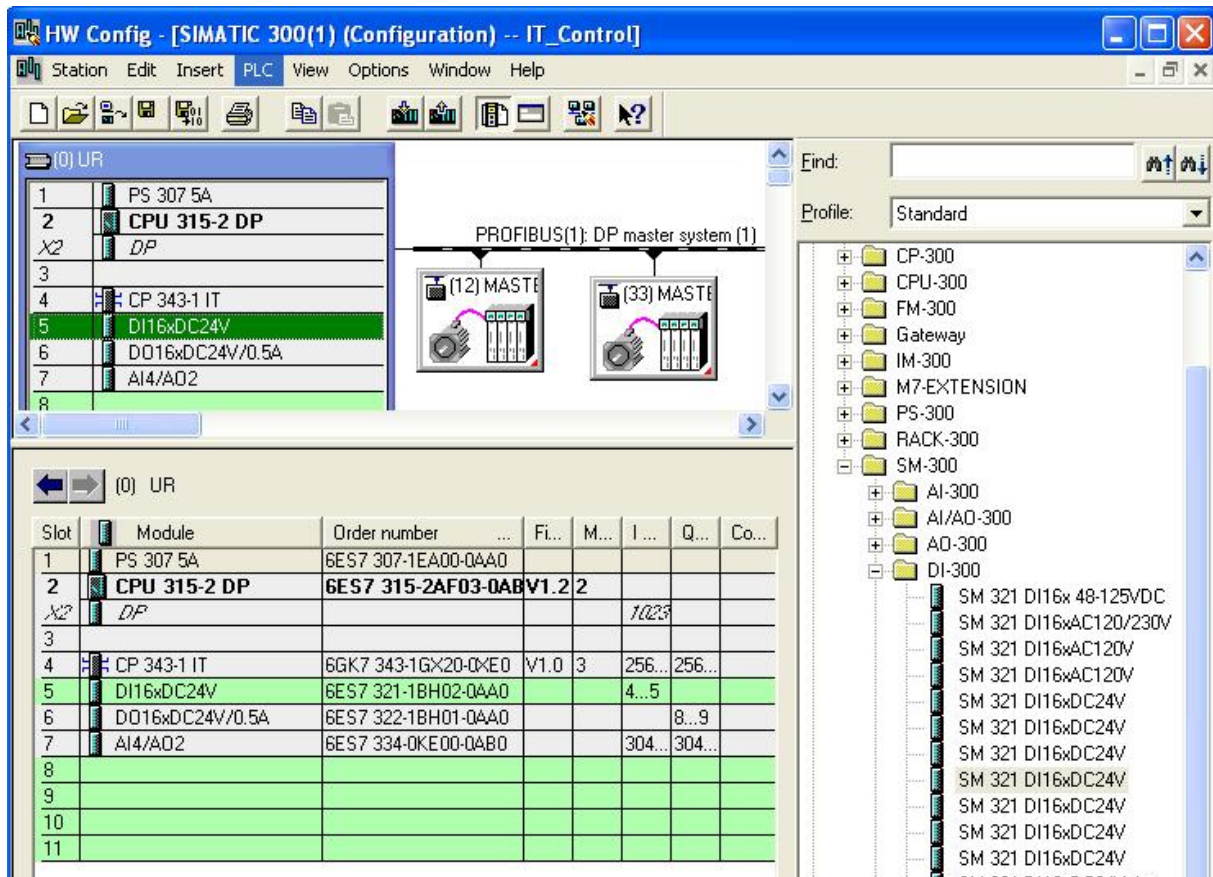


Figure A.16 Insert Digital Inputs (DI)


13. Insert Digital Output module SM 322-1BH01-0AA0

SIMATIC 300 → SM-300 → DO-300 → SM 322-1BH01-0AA0

14. Configure Analogue I/O module

SIMATIC 300 → SM-300 → AI/AO-300 → SM 334-0KE00-0AB0

15. In order to save and compile the project; the hardware configuration must be saved and

compiled by clicking on the  button.

A.2.2 Adding SIEMENS Simovert Masterdrives Motion Controllers

1. Insert Masterdrive MC 12: right click on PROFIBUS(1): DP master system(1)→Insert Object→SIMOVERT→MASTERDRIVES CBP→Motion Control Plus

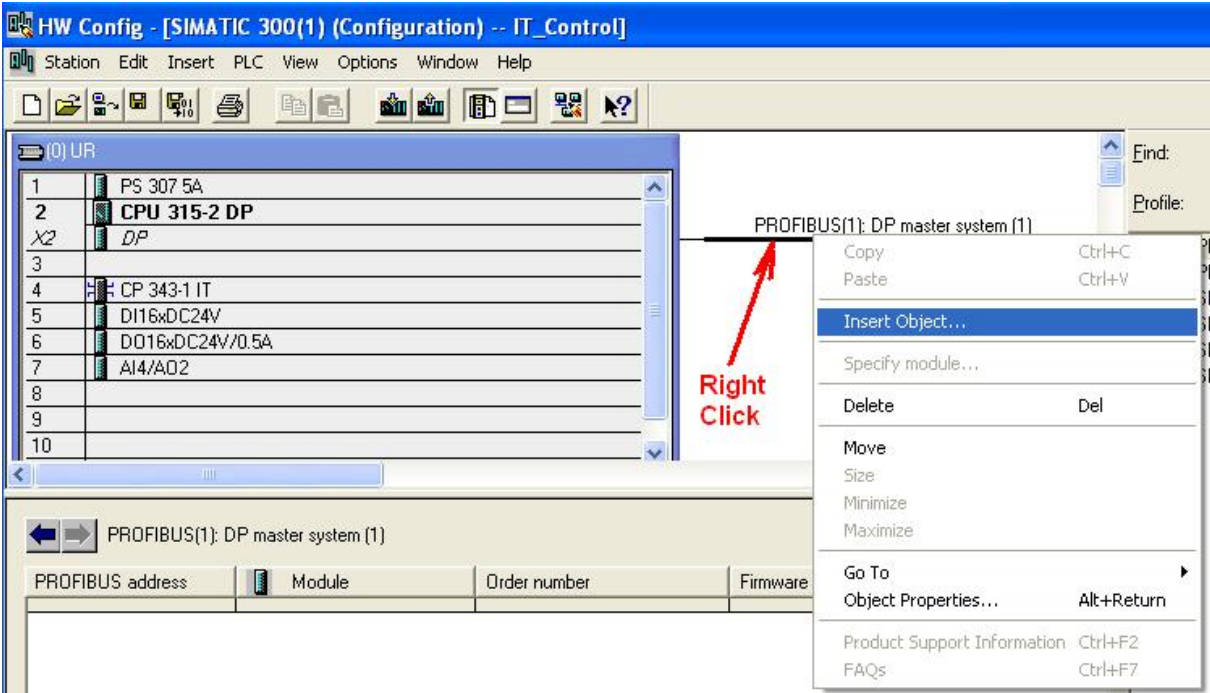


Figure A.17

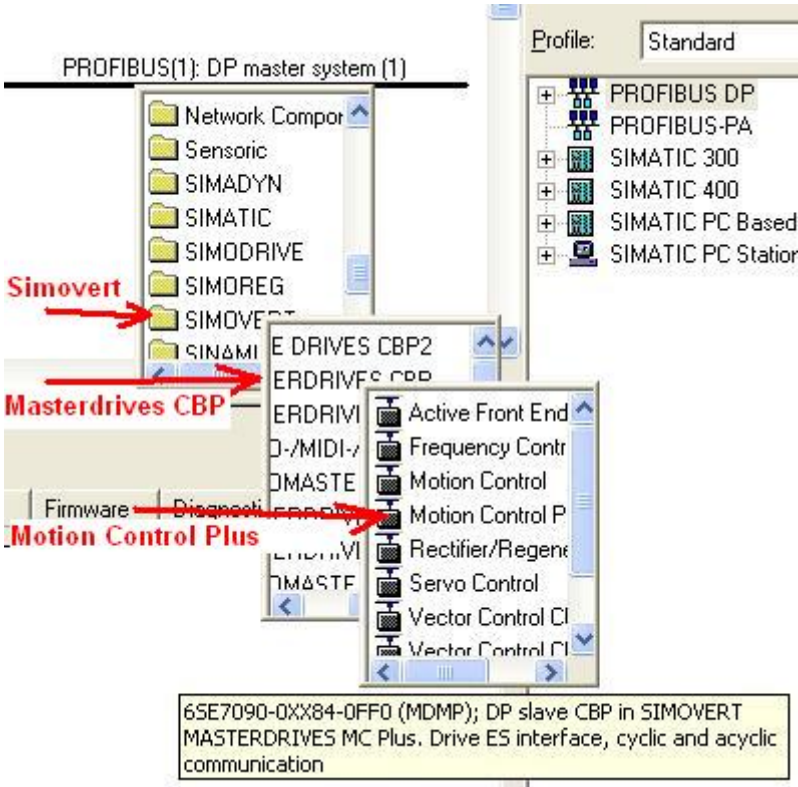


Figure A.18 Add Drivers

2. Set the PROFIBUS bus address: After choosing “Motion Control Plus” at step 22 a window will open and ask for the bus address. One can address up to 128 components with PROFIBUS DP. Here I am choosing address 12. When we press “OK” another window will show up asking for Device version. This is the firmware version of the Motion Controller; in this case it is 1.6.

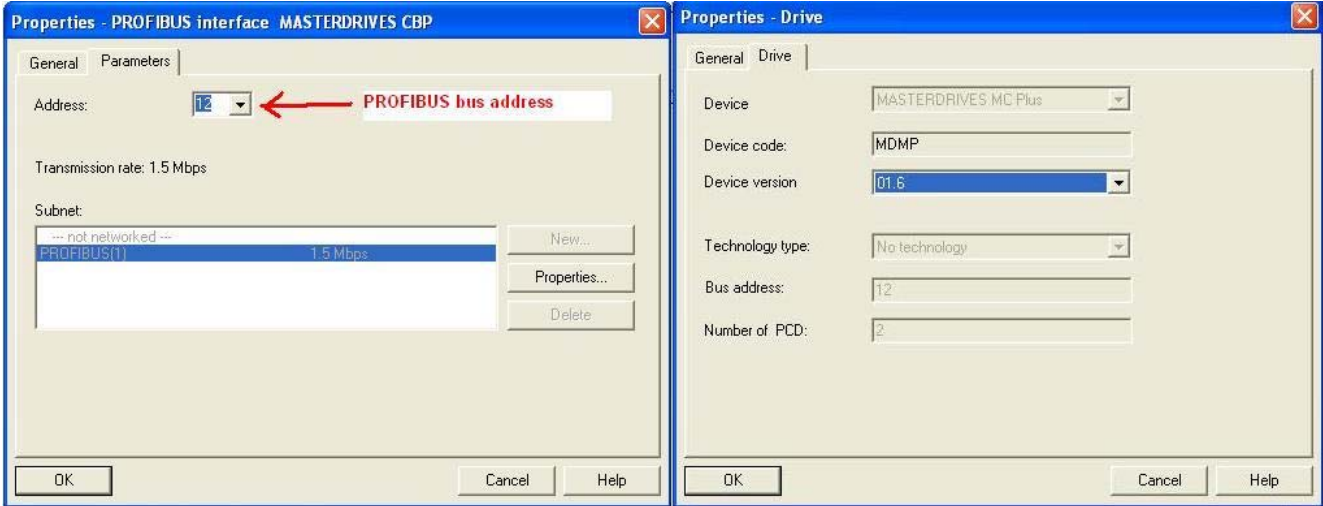


Figure A.19 configure PROFIBUS

3. Set PPO type:
After step 23 there will be another window showing addresses of the Motion controller for writing and reading parameters and “Main Setpoint”. Depending on application and the controllers one has to decide the PPO types. In this case PPO 1 is chosen.

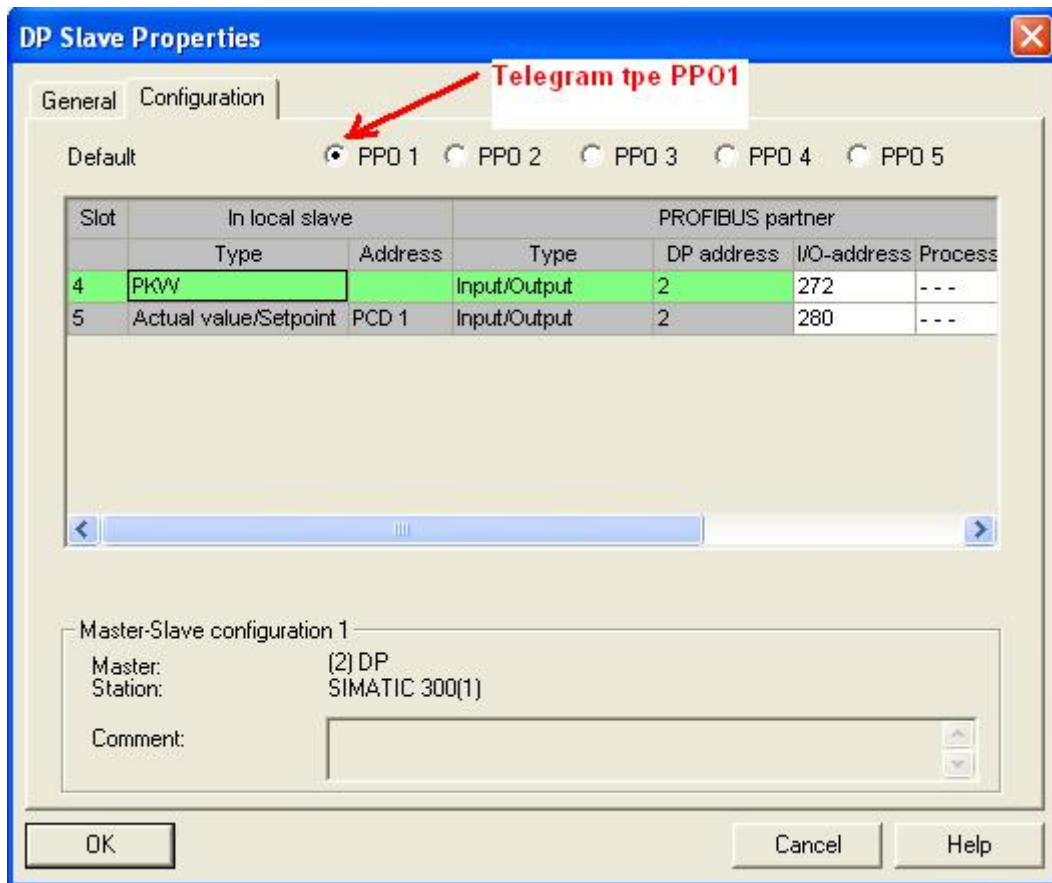


Figure A.20 Configure PPOs

4. Insert Masterdrive MC 33: follow steps 22, 23, and 24, but use 33 as PROFIBUS bus address.
5. Save the configuration then close the Hardware Configuration window.

B Function Diagrams

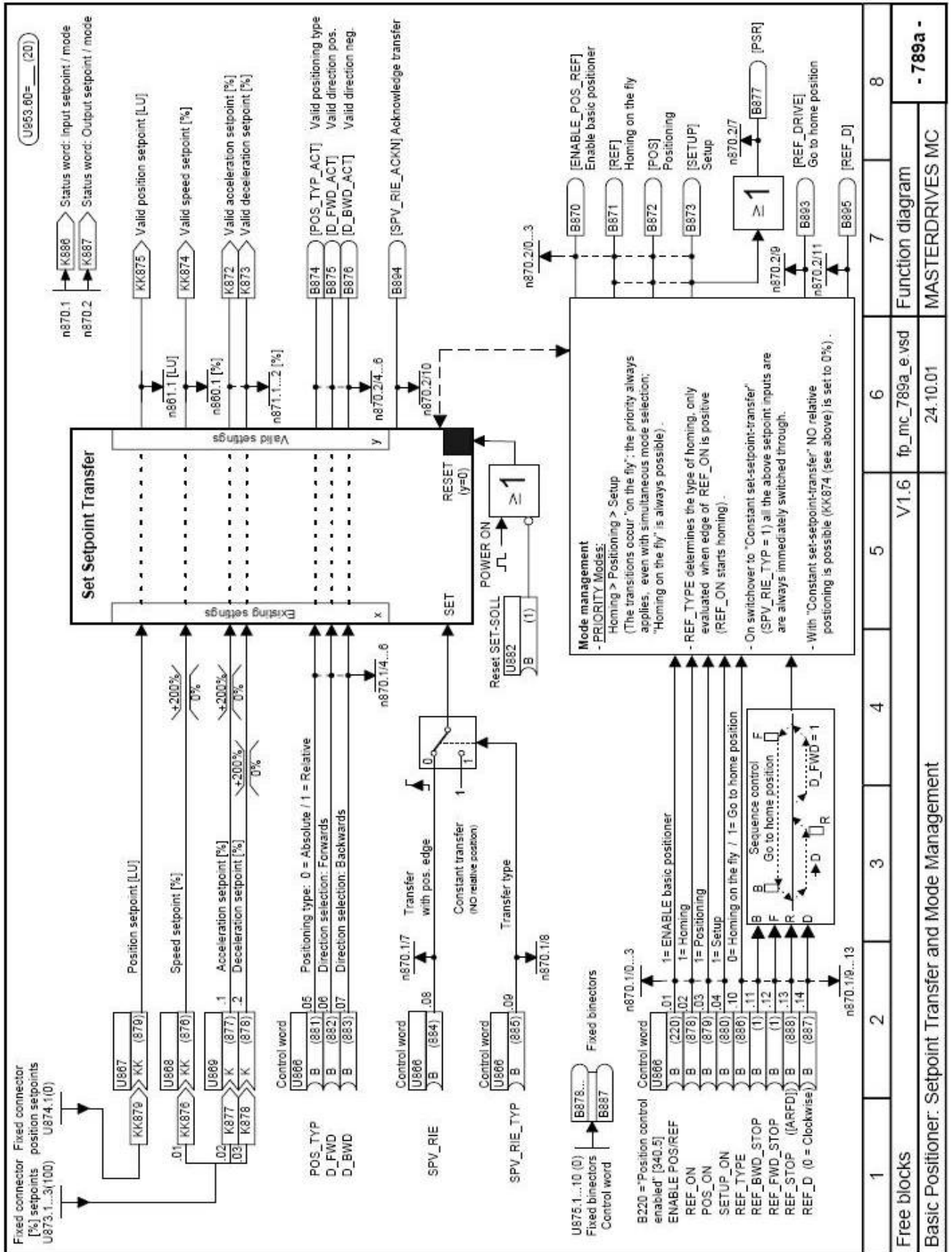


Figure B.1 FD789a[4]

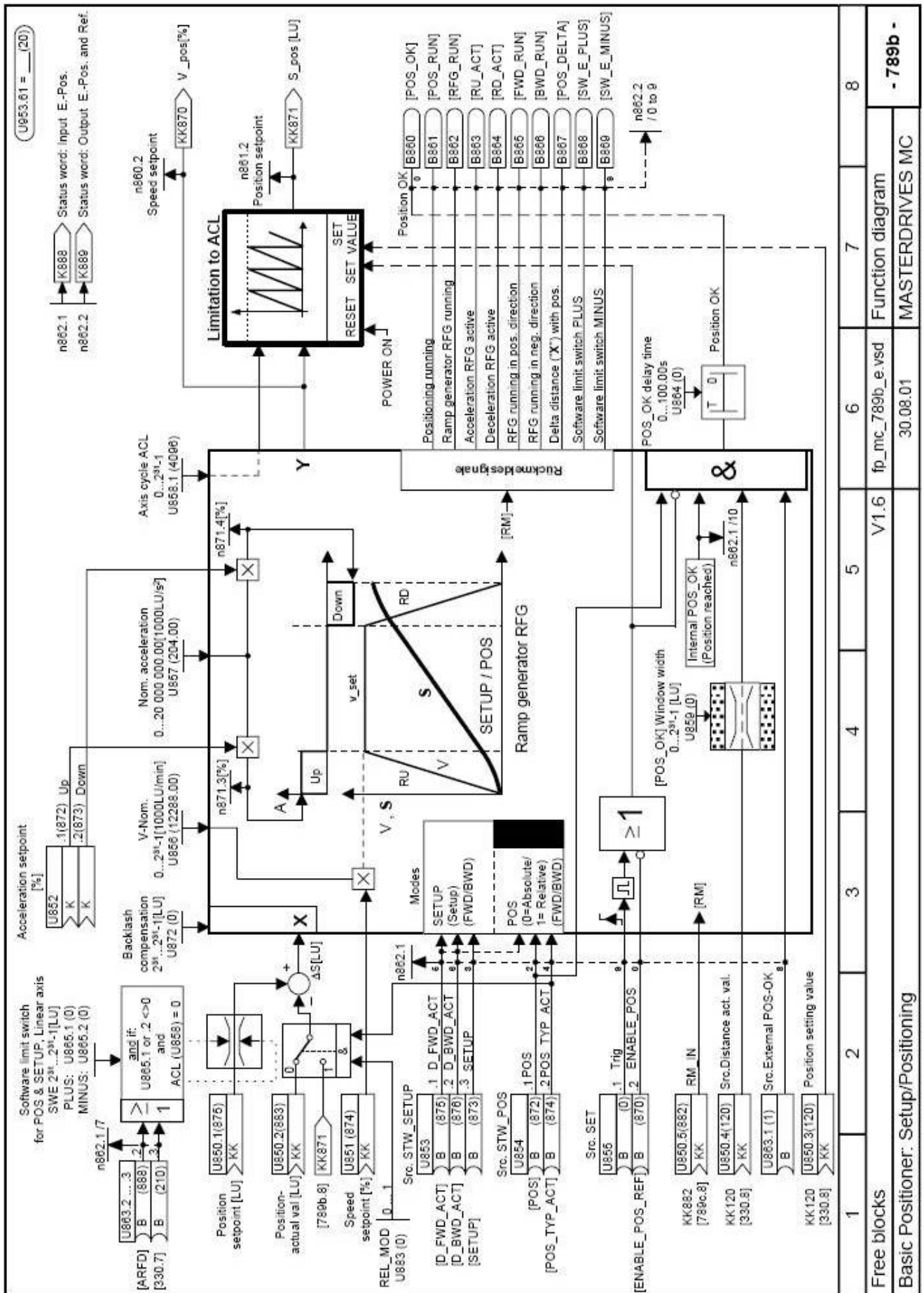


Figure B.2 FD789b[4]

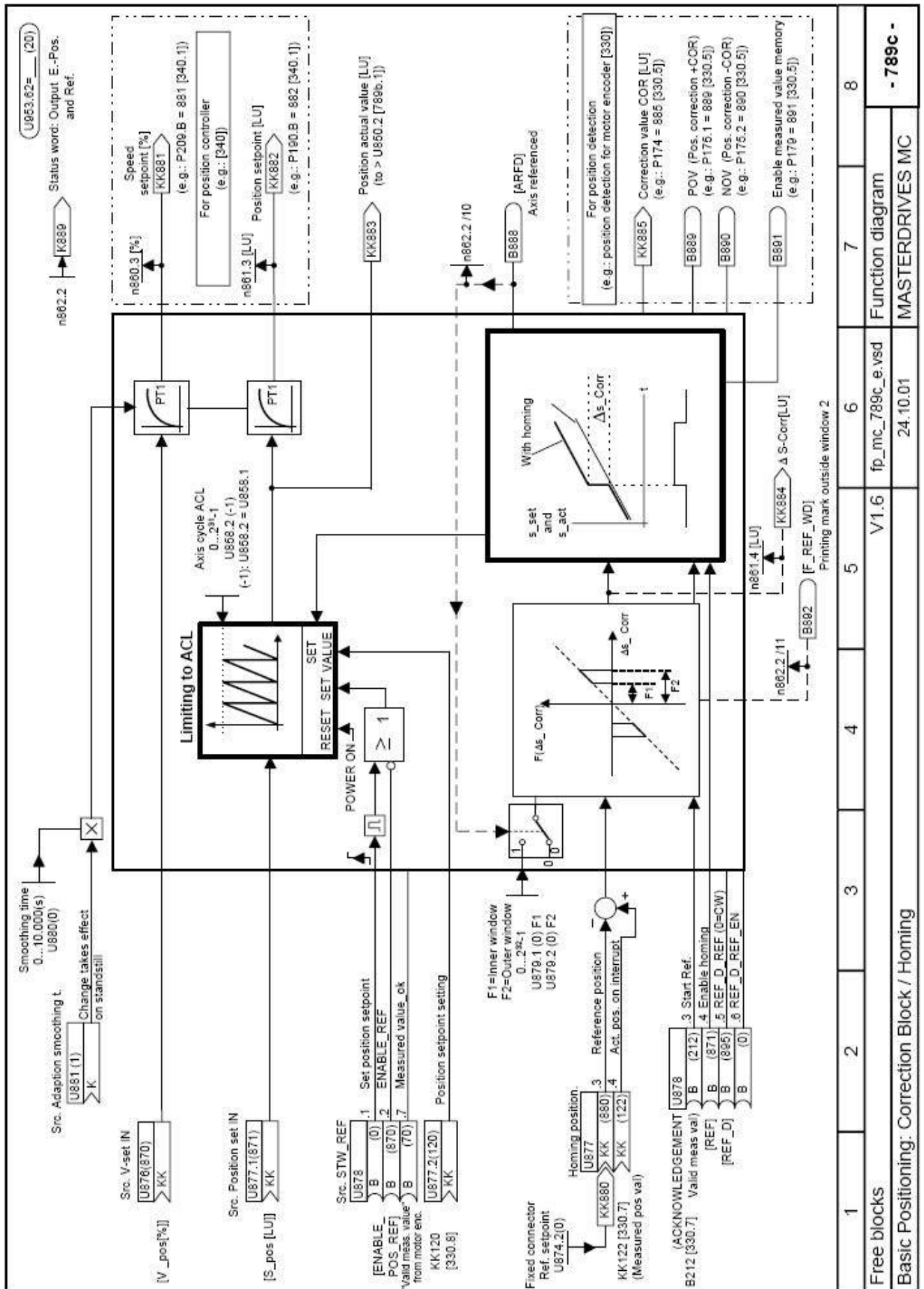


Figure B.3 FD789c[4]

Enabling of the "F01 technology option" (positioning and synchronization)

It is necessary that the F01 technology option has been enabled:

The F01 technology option can only be used with MASTERDRIVES MC units which have been delivered ex-works with the enabled F01 option or for which this option has been enabled retrospectively by means of a PIN number. The display parameter n978 can be used to check if the F01 is present:

```
n978 = 2 ==> Technology option F01 is enabled for 500 h (from V1.31)
n978 = 1 ==> F01 technology option has been permanently enabled
n978 = 0 ==> F01 technology option has been disabled
```

The technology function remains enabled even after a software update and does not have to be entered again after new software has been loaded into the flash EPROM. From version V1.31 onwards, temporary enabling is indicated by the value 2 in n978.

Retrospective enabling of the F01 technology function (involves extra costs):

Proceed as follows if you want to permanently enable the F01 technology option retrospectively

1) Determine the factory serial number of the MASTERDRIVES unit electronics. There are two ways of doing this:

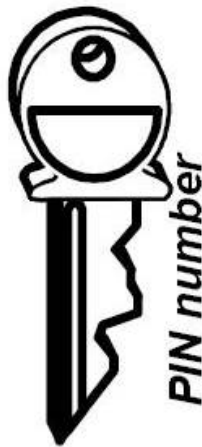
- From parameters U976.01 and U976.02, you can read out the last 8 figures of the factory serial number which are necessary for determining the PIN number.
(Example: U976.01 = 3032, U976.02 = 4198 ==> Factory serial number = ...30324198)
- The serial number can also, if necessary, be obtained from a MASTERDRIVES unit without connecting it to the supply.
- In the case of Compact PLUS units, it is on the sheet accompanying the delivery note or on the electronics PC board in the unit (remove side cover), e.g. "RFU80962510106"
- In the case of Compact and chassis-type units, it is on the upper connector strip on the rear of the CUMC basic electronics board, e.g. "06970730324198"

- Contact your nearest Siemens branch in order to purchase the PIN number which matches your serial number. Quote the last eight figures of the serial number.
 - After you have obtained the PIN number, enter it in parameters U977.1 and U977.2.
 - Switch off the power supply to the electronics and then switch it on again.
 - The F01 technology option is now enabled. You can check this by referring to n978 (see above)
- Caution: If the PIN-No. U977 is subsequently changed, enabling of the technology is reset (n978=0).

Temporary enabling of the F01 technology option (free of charge):

For all units and electronics boards, the F01 technology option can be enabled free of charge with a special PIN No.. This can be done for a trial period of 500 hours. This time can be used for testing purposes or for using substitute units which have been ordered without the F01 option as long as the PIN number has not yet been received. The operating-hours counter (r825) determines when this time has expired. Only that time is counted during which the drive is on. After the 500 hours have expired and the voltage supply has been turned off, the F01 option is disabled again unless the "normal" PIN has been entered in the meantime. The 500 hours can no longer be interrupted (e.g. by changing the PIN entries).
Input of the special PIN number can only be done via the PMU. The special PIN is the same for all units and is as follows

U977.1 = 0727, U977.2 = 0101



Special PIN number

Enabling for units with software version 1.1 (free of charge):

In the case of units which have been supplied with software version V1.1, an individual serial number has not been explicitly entered. In this case, permanent enabling of the F01 technology option when your equipment is upgraded with a new software version is always possible. If you have version V1.1, the following table shows the 4 possible serial numbers and the matching PIN numbers which can be read out in U976 in order to enable the function.

FID		PIN		FID		PIN	
U976.1	U976.2	U977.1	U977.2	U976.1	U976.2	U977.1	U977.2
0000	0000	==>	9970	FFFF	FFFF	==>	9978
0000	FFFF	==>	6662	0000	2800	==>	1970
			5525				0025
			5525				5543

1	2	3	4	5	6	7	8
Technology option							
Enabling with PIN number							
				V1.6		fp_mc_850_e.vsd	
				08.01.02		MASTERDRIVES MC	
						Function diagram	
						- 850 -	

Figure B.4 FD850 [14]

C Instructions to start HMI from the Internet

The page is saved in SIMATIC NET CP 343-1 IT, in order to access the page it should be turned on, and then below steps of instructions need to be followed;

- Use this address in internet browser <http://141.22.15.92/MyProjekt/> you will see the below screen:

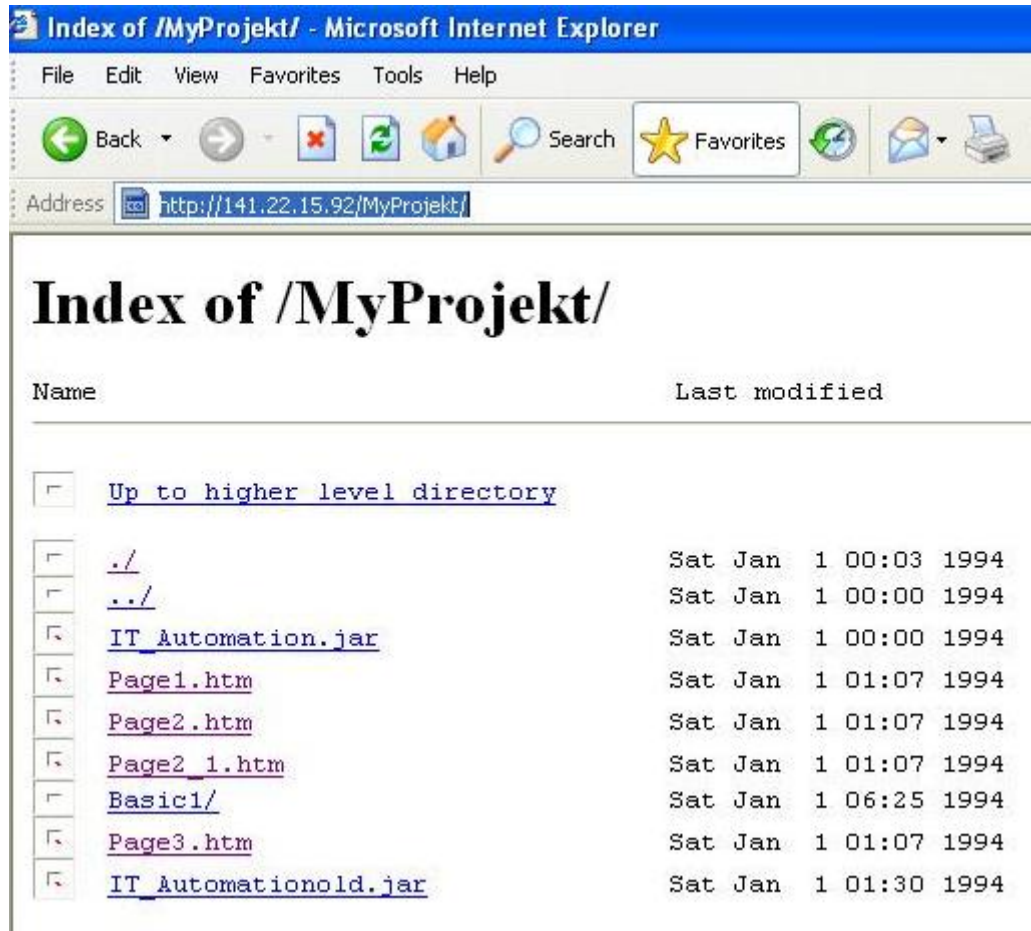


Figure C.1 Index page

- Start (double click on) Page1.htm
You will be prompt blow page (Figure C.2 Login page):

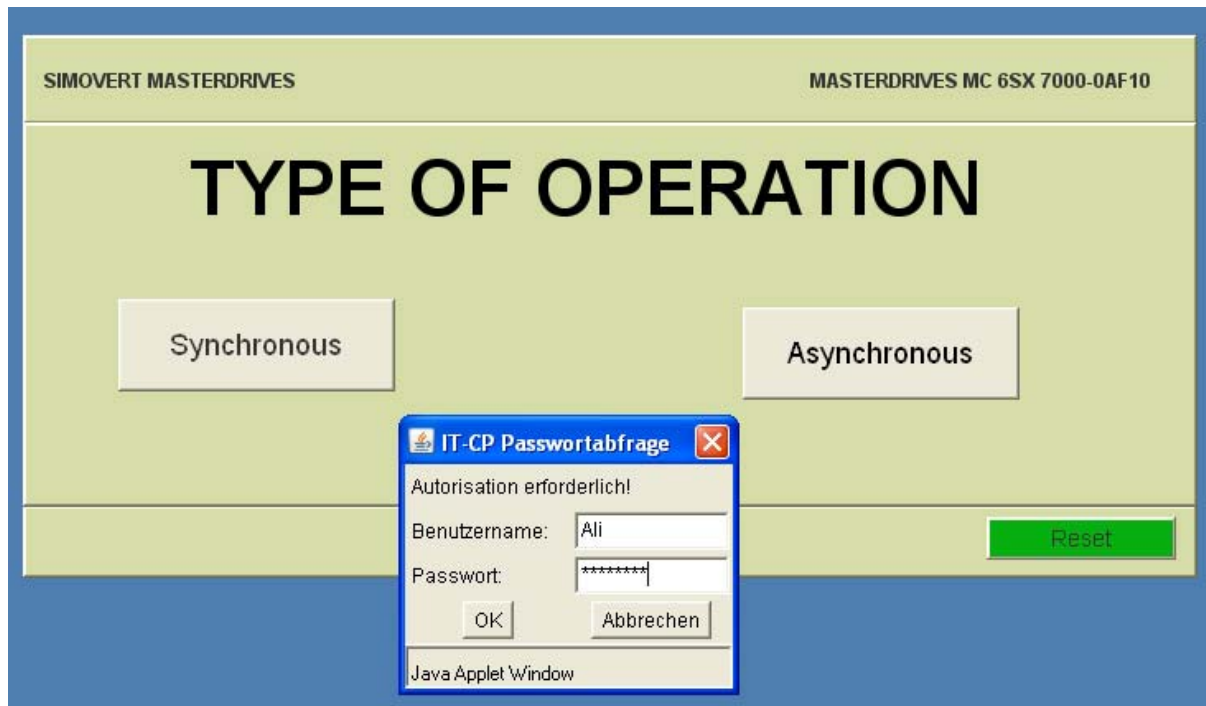


Figure C.2 Login page

- **Username:** Ali (A is capital)
- **Password:** 12345678
- Press “OK”
- Choose the type of operation; “Synchronous” or “Asynchronous”

- If it is Synchronous operation; set the mode you want. For this there is table on the GUI to inform the user to choose any mode he or she wants.
- First turn the right hand side on (Master); DI-1, DI-6, and then DI-5
- Then turn the left hand side on (Slave); DI-1, DI-6, and then DI-5
- If any side does not work; turn DI-5 and DI-6 OFF then turn DI-3 on then off again. After this turn DI-6 and DI-5 on

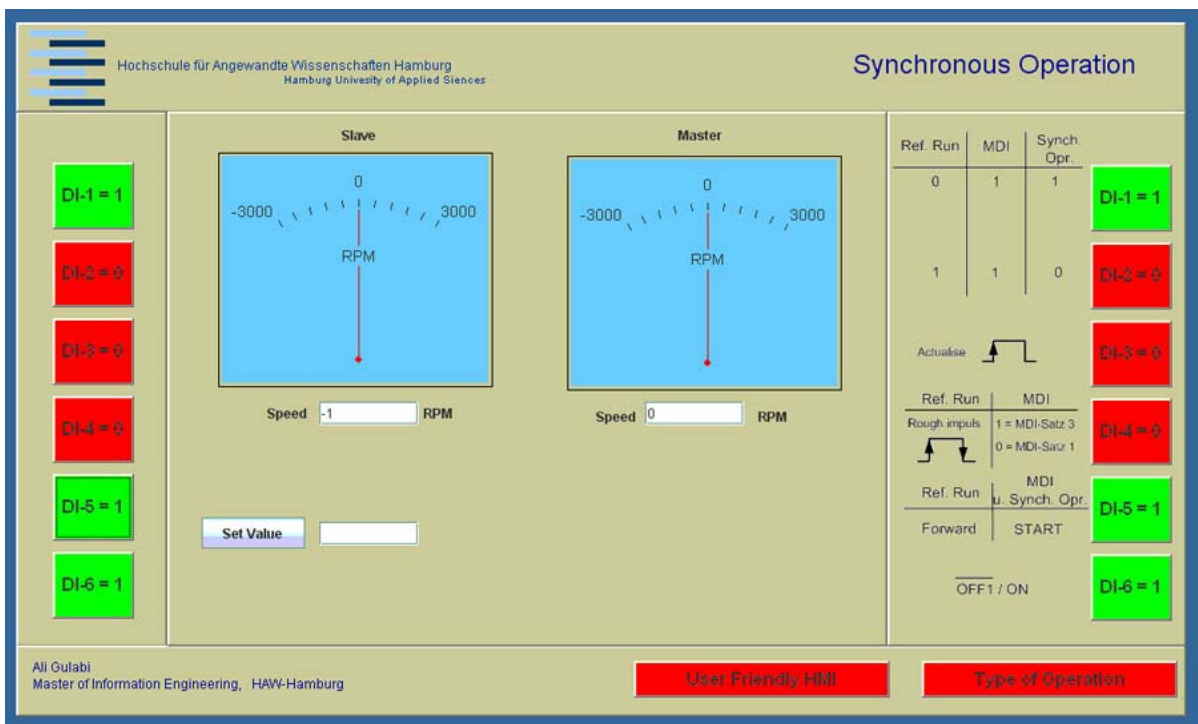


Figure C.3 Synchronous Operation

- If it is too complicated to operate synchronous mode then choose “User Friendly HMI”, where it is more straightforward.
- Press the mode you want
- Press “ON” button
- Press “Start” button

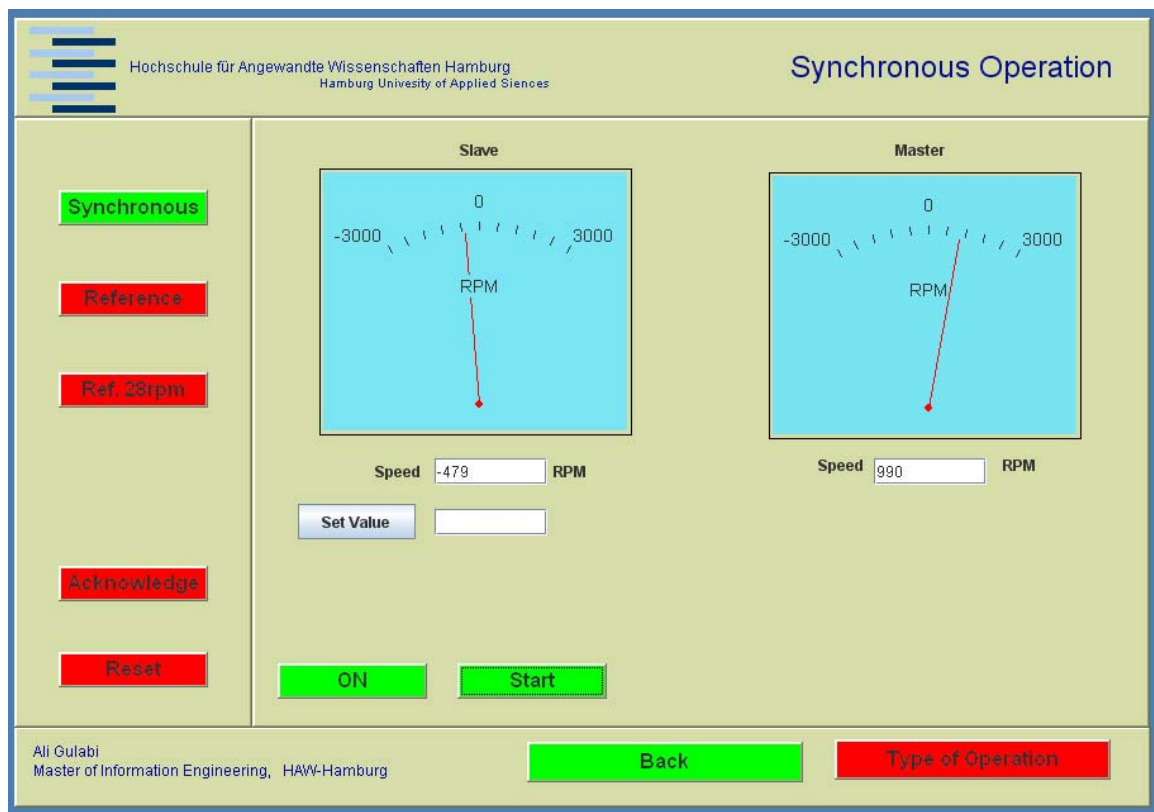


Figure C.4 User friendly HMI

- If motor do not start then; press “Stop”
- Press “OFF”
- Press “Acknowledge” button
- Now press “ON” and “Start” buttons again.
- The speed can be changed.

➤ Asynchronous operation

- You can start Asynchronous operation from the first page. If another operation is currently in operation then stop it and then press “Type of Operation” button for a second.
- Now you are at the first page, and ready to start the Asynchronous operation.

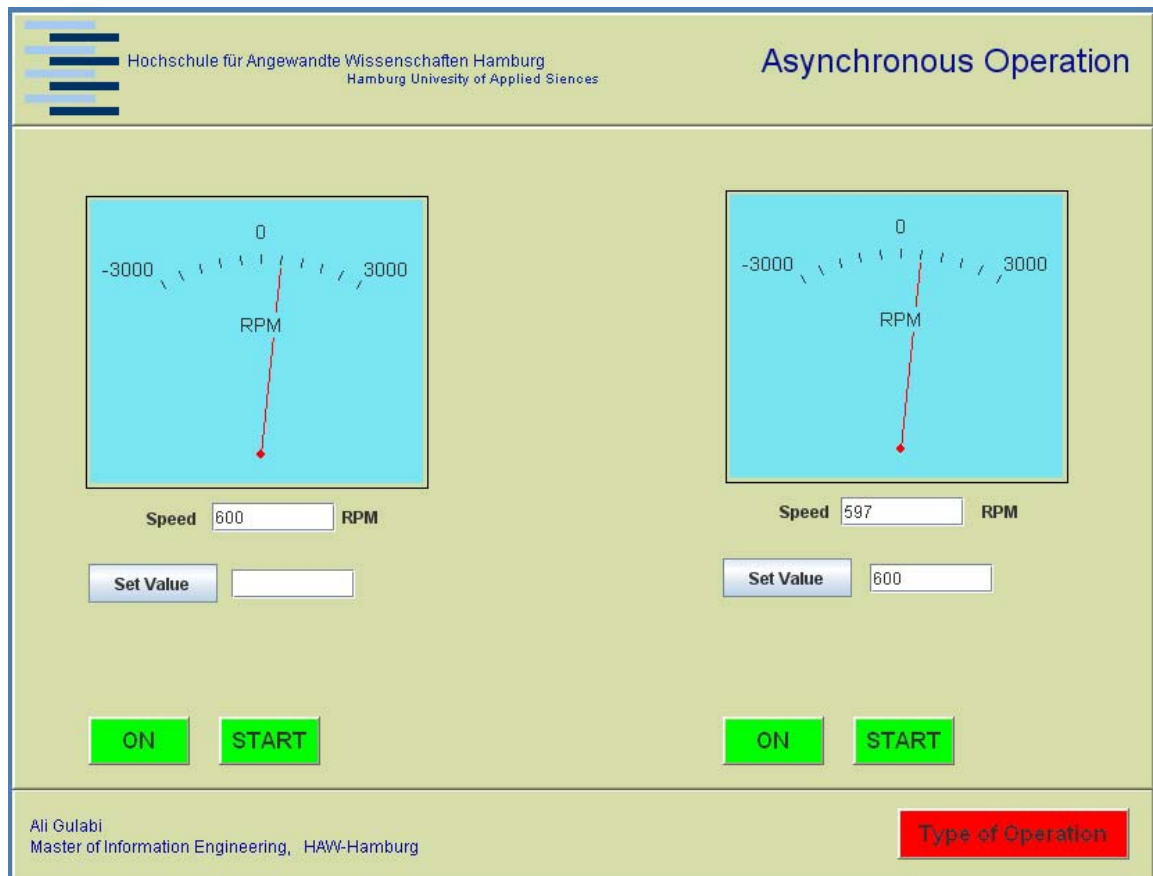


Figure C.5 Asynchronous Operation

- Press “ON” button
- Press “Start” button
- Each side can independently be operated.

D CD-ROM

This Master report contains an appendix of program listing:

- S7- Program including the S7-project
- Java source files
 - Page1.html (Type of Opeation)
 - Page2.html (Synchronous Operation)
 - Page2_1.html (Synchronous Operation use friendly HMI)
 - Page3.html (Asynchronous Operation)
 - IT_Automation.jar (executable jar file)
- Thesis.pdf; a copy of Master Thesis as a PDF file

This appendix is deposited with Prof. Dr. Ing Gustav Vaupel.

E List of Figures

FIGURE 3.1 THE SYSTEM DESCRIPTION	10
FIGURE 4.1 SIMOVERT MASTERDRIVERS MC	11
FIGURE 4.2 SIMATIC S7-300	12
FIGURE 5.1 CP5613 A2	14
FIGURE 5.2 PROFIBUS CONNECTOR	15
FIGURE 5.3 PROFIBUS CONNECTIONS	17
FIGURE 5.4 POWER SUPPLY 380 V	20
FIGURE 5.5 POWER SUPPLY 220 V	20
FIGURE 6.1 A SCREEN SHOT OF S7 COMMUNICATION MODULE	21
FIGURE 6.2 INDUSTRIAL COMMUNICATION PYRAMID [2]	22
FIGURE 6.3 MEANS OF COMMUNICATIONS WITH MASTERDRIVERS MCS	23
FIGURE 6.4 CONNECTOR FOR MPI BUS	24
FIGURE 6.5 A SAMPLE VIEW OF PROFIBUS	25
FIGURE 6.6 INDUSTRIAL ETHERNET	27
FIGURE 7.1 SIMOVERT MASTERDRIVES MC	29
FIGURE 7.2 FREE BLOCKS [3]	30
FIGURE 7.3 FUNCTION BLOCKS	32
FIGURE 7.4 CONNECTORS AND BINECTORS	33
FIGURE 7.5 CONNECTING TWO FUNCTION BLOCKS	33
FIGURE 7.6 PARAMETER MENUS	34
FIGURE 7.7 PMU	35
FIGURE 7.8 OP1S	36
FIGURE 7.9 OP1S SCREEN SHOTS	37
FIGURE 7.10 DERIVE PROPERTIES	38
FIGURE 7.11 DRIVE MONITOR COMMUNICATION	39
FIGURE 7.12 SET SERIAL PORT BUS ADDRESS	39
FIGURE 7.13 PARAMETER ACCESS	40
FIGURE 7.14 DRIVE MONITOR ONLINE BUTTON	40
FIGURE 7.15 PROJENCT INSERT PARAMETER SET	41
FIGURE 7.16 DRIVE MONITOR DRIVE NAVIGATOR	42
FIGURE 7.17 DETAILED PARAMETERIZATION STEPS	45
FIGURE 8.1 GEARS	49
FIGURE 8.2 THREE FREE BLOCKS	50
FIGURE 8.3 LINEAR AXIS [3]	52
FIGURE 8.4 ROTARY AXIS	52
FIGURE 8.5 ROLL FEED [3]	52
FIGURE 8.6 DESCRIPTION OF THE APPLICATION	54
FIGURE 8.7 DIGITAL INPUTS (TERMINAL STRIP DIAGNOSTICS)	55
FIGURE 9.1 CBP COMMUNICATIONS BOARD (COMMUNICATIONS BOARD PROFIBUS)	58
FIGURE 9.2 DATA TRAFFIC CHANNELS OF CBP [4]	59
FIGURE 9.3 PARAMETER IDENTIFIER VALUE	60
FIGURE 9.4 PPO TYPES	60
FIGURE 9.5 PPO OVERVIEW IN HARDWARE CONFIGURATION	61
FIGURE 9.6 SFC 15 WRITING DATA TO DP SALE	63
FIGURE 9.7 PZD1 (CONTROL WORD 1)	65
FIGURE 9.8 DIAGNOSTICS TOOL READ/WRITE STATUS AND CONTROL WORDS	66
FIGURE 10.1 FLOW DIAGRAM OF S7 CONTROL PROGRAM	68
FIGURE 10.2 FLOW DIAGRAM OF CONTROL PROGRAM	69
FIGURE 10.3 COMMON PROGRAM FLOW	70
FIGURE 10.4 PARAMETERIZATION FLOW DIAGRAM	71
FIGURE 10.5 DB85 DATABASE USED FOR MC 12	73
FIGURE 10.6 VALUE RANGE OF 16-BIT CONNECTORS (KXXXX) [1]	76
FIGURE 10.7 INPUT AND OUTPUT ADDRESSES OF PERIPHERALS	77
FIGURE 10.8 PZD1 CONTROL WORD 1	81
FIGURE 11.1 ADD EXTERNAL JARS	86
FIGURE 11.2 IMPORT EXTERNAL JARS TO THE PROJECT	87
FIGURE 11.3 THE PROJECT WINDOW WITH APIS	87
FIGURE 11.4 THE STRUCTURE OF TASK	88

FIGURE 11.5 API LIBRARY V2.5.5-----	89
FIGURE 11.6 COMPARISON BETWEEN HW AND S7-BEANS-----	90
FIGURE 11.7 JAVA VIRTUAL MACHINE [10]-----	92
FIGURE 11.8 VISUAL PROJECT IN ECLIPSE-----	97
FIGURE 11.9 ADDING BEANS FROM THE PALETTE-----	99
FIGURE 11.10 DEVICE API INSERTED THROUGH PALETTE-----	100
FIGURE 11.11 HMI PAGE1-----	101
FIGURE 11.12 COMPONENTS HIERARCH-----	102
FIGURE 11.13 A SCREEN SHOT OF PAGE2 OF GUI-----	105
FIGURE 11.14 USER FRIENDLY HMI-----	111
FIGURE 11.15 HMI ASYNCHRONOUS OPERATION-----	113
FIGURE A.1 STARTING PG-PC INTERFACE-----	118
FIGURE A.2 ACTIVATE MPI-----	119
FIGURE A.3 CREATE A NEW PROJECT-----	120
FIGURE A.4 ENTERING HARDWARE COMPONENTS-----	120
FIGURE A.5 ENTER SIMATIC 300 STATION-----	121
FIGURE A.6 ENTER INDUSTRY ETHERNET-----	121
FIGURE A.7 HARDWARE CONFIGURATION-----	122
FIGURE A.8COMPONENT CATALOGUE-----	122
FIGURE A.9 INSERT RAIL-----	123
FIGURE A.10 ADD POWER SUPPLY-----	123
FIGURE A.11 INSERT CPU-----	124
FIGURE A.12 MPI PROPERTIES-----	124
FIGURE A.13 ADD COMMUNICATION PROCESSOR-----	125
FIGURE A.14 SET IP ADDRESS-----	125
FIGURE A.15 ADD USERS-----	126
FIGURE A.16 INSERT DIGITAL INPUTS (DI)-----	127
FIGURE A.17-----	128
FIGURE A.18 ADD DRIVERS-----	128
FIGURE A.19 CONFIGURE PROFIBUS-----	129
FIGURE A.20 CONFIGURE PPOS-----	130
FIGURE B.1 FD789A[4]-----	131
FIGURE B.2 FD789B[4]-----	132
FIGURE B.3 FD789C[4]-----	133
FIGURE B.4 FD850 [4]-----	134
FIGURE C.1 INDEX PAGE-----	135
FIGURE C.2 LOGIN PAGE-----	136
FIGURE C.3 SYNCHRONOUS OPERATION-----	137
FIGURE C.4 USER FRIENDLY HMI-----	138
FIGURE C.5 ASYNCHRONOUS OPERATION-----	139

F List of Tables

TABLE 6.1 PROFIBUS TRANSMISSION RATES	26
TABLE 6.2 TYPES OF COMMUNICATIONS FOR S7 300/400	28
TABLE 7.1 SIMOVERT MASTERDRIVES MC SPECIFICATIONS	30
TABLE 7.2 POWER SECTION	46
TABLE 7.3 BOARD CONFIGURATION	46
TABLE 7.4 BOARDS TO BE CONFIGURED	46
TABLE 9.1 FIXED ASSIGNMENT AND COMBINATION VALUES	62
TABLE 9.2 TELEGRAM STRUCTURE PKE AREA	62
TABLE 9.3 TASK ID MASTER TO CONVERTER	63
TABLE 9.4 CONNECTION DI TO PZD1	65
TABLE 10.1 BLOCK TYPES	67
TABLE 10.2 TELEGRAM PKW AREA	70
TABLE 10.3 PKW AND PZD PERIPHERAL ADDRESSES	77
TABLE 10.4 S7 FUNCTIONS TO WRITE AND READ EXTERNAL INPUT AND OUTPUTS	78
TABLE 10.5 CONTROL WORD IN BIG-ENDIAN STRUCTURE	79
TABLE 10.6 DI ASSIGNMENTS	79
TABLE 11.1 JAVA APPLET	93
TABLE 11.2 S7-BEANS FOR DEVICES	94
TABLE 11.3 S7-BEANS FOR GUI	95
TABLE 11.4 S7 UTILITY BEANS	96
TABLE 11.5 PARAMETER VALUES OF THE S7ANYPOINTER	103
TABLE 11.6 CONTROL BUTTONS AND ADDRESSES	109

Declaration

I declare within the meaning of section 25(4) of the Examination and Study Regulations of the International Degree Course Information Engineering that: this Master report has been completed by myself independently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or works of others are made known through the definition of sources.

City

Date

signature