

## Integrated Monitoring and Control System

# IMAC L

### Engineering Guideline:

- Structuring of the System
- Compilation of Measuring Points Lists
- MePo Parameter Descriptions
- Building an IMAC L System (code generating)
- S7 Programming of IMAC L Functions
- S7 Programming of Process Control Functions
- HMI System
- Setting of dynamic Parameters for the HMI System

**Rev. G**

Some hints to navigate within this document:

If this document is available as a file, you can use the implemented hyperlinks for easy navigation. Just point with the mouse to an entry in the table of contents or to a hyperlink and click. To “jump” back, use the keys “Alt + ←”, to “jump” forward use “Alt + →”

## Table of Contents

A.	General References.....	7
1.	Introduction .....	8
1.1	About this description.....	8
1.2	Symbols and conventions .....	8
2.	Basic Structure of IMAC L .....	9
2.1	Technology Areas, SECTION DATA and Process Graphics Displays .....	9
2.1.1	Structuring, Identification System.....	10
2.1.2	Structuring, Identification for Measuring Points.....	11
2.1.3	Use of Identification Letters in MePo-Ids to identify internal Measuring Points .....	12
2.1.4	Structuring, Identification for Process Graphics Displays (GDs) .....	15
2.1.5	Structuring, Identification for Section Data .....	15
3	Measuring Points Lists - Basics .....	16
3.1	Measuring Point Description Texts .....	16
3.1.1	Engineering Guideline.....	16
3.1.2	Basic Abbreviations List (English / German) for Measuring Point Description Texts .....	17
3.1.3	Basic Abbreviations List (Englisch / German) for Measuring Point Status Texts .....	19
4.	IMAC L Measuring Points Lists Front-end .....	24
4.1	Opening the Excel MePo Front-end .....	25
4.2	Worksheets of the IMAC L MePo Front-end.....	26
4.2.1	Worksheet 'Config' .....	26
4.2.2	Worksheet 'Objekte' .....	27
4.2.3	Worksheet 'Status Texts' .....	31
4.2.4	Worksheet 'Status' .....	32
5.	Creating and Modifying MePos with the 'Edit' Form .....	33
5.1	Basic Editing Functions.....	34
5.2	Fault Indications in the Edit Form .....	35
5.3	Description of all Input Fields in the Edit Form .....	36
5.3.1	Field : 'PointID' .....	36
5.3.2	Field : 'Obj-Class' .....	36
5.3.3	Field : 'MP-Type' .....	37
5.3.4	Field : 'PV-Type' .....	37
5.3.5	Field : 'Description' .....	37
5.3.6	Field : 'Station' .....	38
5.3.7	Field : 'Section Data' .....	38
5.3.8	Field : 'TechID' .....	39
5.3.9	Field : 'AiPi' .....	39
5.3.10	Field : 'AiVi' .....	40
5.3.11	Field : 'HiTi' .....	40
5.3.12	Field : 'UndefGrp' .....	41
5.3.13	Field : 'FNoAckn' .....	41
5.3.14	Field : 'FNoChange' .....	42
5.3.15	Field : 'EVL' .....	42
5.3.16	Field : 'EVP' .....	43
5.3.17	Field : 'Address' .....	43

5.3.18	'Smart Help' Button : '?' .....	43
5.3.19	Field : 'Value (default)' & 'InitValue' .....	45
5.3.20	Field : 'Transmitter' .....	45
5.3.21	Field : 'Terminal' .....	45
5.3.22	Field : 'Format' .....	46
5.3.23	Field : 'Range Start' .....	46
5.3.24	Field : 'Range End' .....	46
5.3.25	Field : 'Unit' .....	47
5.3.26	Field : 'Send Hyst' .....	47
5.3.27	Field : 'Limit Hyst' .....	48
5.3.28	Field : 'CompID' .....	48
5.3.29	Field : 'ScalGroup' .....	49
5.3.30	Field : 'No Of Status' .....	49
5.3.31	Fields : 'A Lvl' .....	50
5.3.32	Fields : 'BLG' .....	51
5.3.33	Fields : 'VAG' .....	53
5.3.34	Fields : 'AAG' .....	53
5.3.35	Fields : 'SOG' .....	54
5.3.36	Fields : 'Status Text' .....	54
5.3.37	Fields : 'Delay In' & 'Delay Out' .....	55
5.3.38	Fields : 'Limit(s)' .....	55
5.3.39	Fields : 'Dyn Limit Group' .....	56
5.3.40	Field : 'Remarks' .....	58
5.3.41	Type Indication NO / NC for Binary MePos .....	58
5.3.42	'Change Type to ...' Button for binary 'NO/NC' MePos' .....	58
5.4	Application Examples for IMAC L Measuring Points .....	59
5.4.1	Binary Measuring Point without Alarm .....	59
5.4.2	Binary Measuring Point with Alarm (NO & NC Type) .....	60
5.4.3	Analogue Measuring Point without Limits .....	62
5.4.4	Analogue Measuring Point with Limits .....	63
6	Simatic S7 Configuration .....	64
6.1	Simatic S7 Hardware Configuration .....	64
6.2	Simatic S7 'NetPro' Configuration .....	67
7	Generating Simatic S7-Code and Vision NT Runtime Data .....	69
7.1	Deleting of old AvET data from previous Excel-Importing .....	69
7.1.1	Deleting of Sections .....	69
7.1.2	Deleting of obsolete Measuring Points .....	70
7.1.3	Deleting of obsolete Process Variables .....	71
7.2	Importing MePo data from Excel .....	72
7.3	Automatic Assigning of Object Numbers .....	74
7.4	Generating the AbvSPU Code for Simatic S7 .....	75
7.5	Generating the VisuNT database files for AlphaVision .....	76
7.6	Copying of DBF-Files for Vision-NT .....	78
7.7	Compiling of AlphaVision Graphics Pictures .....	79
7.8	Copying of Visu Runtime Files (*.DIS / *.DO) .....	82
7.9	Import of AbvSPU Code into the Simatic S7 Manager Software .....	83
7.10	Transfer of Simatic S7 AbvSPU Code to the PCU(s) .....	87
8	Engineering in AvET .....	88
8.1	Blocking Groups .....	88
8.2	Scaling Groups .....	91
8.2.1	EXCEL 'Tank Curve' Tool .....	94
8.3	Dynamic Limit Groups .....	98
8.4	Transmitter Types .....	102
9	Programming Simatic S7-Code for IMAC L Functions .....	106
9.1	Undefined Groups .....	106
9.2	Signal Output Groups .....	107

9.3	Exhaust Gas Mean Value Deviation Computing.....	108
9.3.1	IMAC L Exhaust Gas Temperature Deviation Function.....	109
9.3.2	Parameters required to set up Exhaust Gas Function .....	109
9.3.3	I/O data of Exhaust Gas Function .....	110
9.3.4	Control of Exhaust Gas Symmetrization .....	112
9.3.5	Measuring Points required to set up Exhaust Gas Deviation Alarms .....	112
9.4	Monitoring of Profibus DP Slaves .....	114
9.5	Audible Alarm and 'Horn Stop' .....	116
9.5.1	Alarm Forwarding.....	117
9.6	Application Watchdog for OS Software-Tasks.....	118
9.7	Access to MePo Values and Status from Control Programs .....	121
9.7.1	Automatic Retrieval of IMAC L Object Numbers .....	123
9.8	Time Synchronization & Time Setting in IMAC L.....	126
9.8.1.	Setting of the IMAC L Universal Time (UT) from OS.....	126
9.8.2.	Setting of the IMAC L Local Time (LT) from OS.....	127
9.8.3.	Setting of the IMAC L Universal Time (UT) from PCU .....	127
9.8.4.	Setting of the IMAC L Local Time (LT) from PCU .....	128
9.8.5.	Logging Time & Date setting Events in the EventLog .....	128
9.8.6.	Consequences of Setting the UT time back into the Past .....	128
9.8.7.	Time setting precision between Stations in IMAC L .....	129
9.8.8.	Time Synchronization Strategy between Stations in IMAC L .....	130
9.8.8.	TimeMaster Settings .....	130
9.8.8.1.	TimeMaster Settings in AvEdit .....	130
9.8.8.2.	TimeMaster Settings in VISION2000.INI.....	131
9.8.8.3.	TimeMaster additional System Settings for Upgrade from old Versions .....	131
10	Programming Simatic S7-Code for Process Control .....	134
10.1	Introduction to Process Control Functions.....	134
10.1.1	Software Block Overview .....	135
10.1.2	Special Function Blocks .....	136
10.1.2.1	Delay Timer.....	136
10.1.2.2	Pulse Timer.....	137
10.1.3	Software Structure .....	139
10.1.4	Memory Distribution .....	140
10.1.5	Activation of Template Objects .....	142
10.1.6	Extending the Number of Objects .....	144
10.2	Valves / Flaps.....	152
10.2.1	Used Resources .....	152
10.2.2	Description I/O Area.....	152
10.2.3	Description Static Variables (options) .....	157
10.3	Pumps / Motors .....	158
10.3.1	Used Resources .....	158
10.10.2	Description I/O Area.....	158
10.3.3	Description Static Variables (options) .....	163
10.4	Pumps / Motors (2 Speeds OR 2 Directions).....	164
10.4.1	Used Resources .....	164
10.4.2	Description I/O Area.....	164
10.4.3	Description Static Variables (options) .....	170
10.5	Pumps / Motors (2 Speeds AND 2 Directions) .....	171
10.5.1	Used Resources .....	171
10.5.2	Description I/O Area.....	171
10.5.3	Description Static variables (options).....	177
10.6	Stand-By Pumps / Motors .....	178
10.6.1	Used Resources .....	178
10.6.2	Description I/O Area.....	178
10.6.3	Description Static variables (options).....	182
10.7	Tanks .....	183
10.7.1	Used Resources .....	183
10.7.2	Description I/O Area.....	183
10.7.3	Description Static Variables (options) .....	185



10.8	Converting SoftPLC7 Programs.....	187
11.	IMAC L Human Machine Interface (HMI).....	192
11.1.	Basic Rules for the IMAC L HMI System.....	192
11.2.	Tips on IMAC L Screen Size / Resolution issues .....	192
11.2.1.	Caption Bar of the Application Area .....	192
11.2.1.	Slider Bars of the Application Area .....	193
11.3.	Tips for Printing with IMAC L .....	194
11.3.1.	Eventlog Printing with IMAC L.....	194
11.3.2.	Report Printing with IMAC L.....	194
11.4	IMAC L 'Hints & Tips' (HiTi) .....	196
11.5.	User Rights and 'Station In Control' Concept .....	200
11.5.1.	Start Batch to run SoftPLC Program for 'OS 99' .....	200
11.5.2.	ReCompile the SoftPLC Program for 'OS 99' .....	201
11.5.3.	Update the AbvSPU of the SoftPLC Program for 'OS 99' .....	201
11.5.4.	Engineering for SIC.....	202
11.5.5.	Engineering for Users .....	204
11.5.6.	Engineering for User Profiles .....	205
11.5.6.1.	IMAC L Function Numbers.....	206
11.6.	Terminating the IMAC L tasks of an OS .....	206
11.7.	Dynamic HMI Elements for Process Control .....	207
11.7.1.	MePos required for HMI access to 'Control Function' Objects .....	208
11.7.2.	System Objects required for each GD.....	209
11.7.3.	Reference Objects required for each GD.....	210
11.7.4.	Dynamic Attributes for 'Indication and Control' HMI Objects.....	212
11.7.4.1.	Starting a new Graphics display.....	212
11.7.4.2.	Editing properties of static text fields.....	213
11.7.4.3.	Setting dynamic Attributes for Routing Buttons.....	214
11.7.4.4.	Setting dynamic Attributes for Process Object Figures (except TANK, STANDBY).....	219
11.7.4.5.	Setting dynamic Attributes for Process Object Figure 'Tank' .....	221
11.7.4.6.	Setting dynamic Attributes for 'STANDBY' Control Buttons .....	223
11.7.4.7.	Setting dynamic Attributes for the Alarm Triangles .....	224
11.7.4.8.	Setting dynamic Attributes for the Alarm Status Indications.....	226
11.7.4.9.	Setting dynamic Attributes for the Control Position Indications.....	227
11.7.4.10.	Setting dynamic Attributes for 'Fan Direction' Indications .....	228
11.7.5.	Setting dynamic Attributes for Control Windows .....	229
11.7.5.1.	Copying of required Control Window Components .....	229
11.7.5.2.	Setting dynamic Attributes of the new Control Window Figures.....	231
11.7.5.3.	Placing the new Control Window in the new GD.....	234
11.7.5.4.	Setting dynamic Attributes of the new Control Window.....	235
11.7.6.	Modifying the Header Area .....	236
11.7.6.1.	Modifying the 'Text Labels' on the Buttons in the Header Area.....	236
11.7.6.2.	Modifying the 'Button Actions' of the Buttons in the Header Area .....	237
11.7.7.	Copying / Modifying the Menu Areas .....	238
11.8.	Ship Specific Settings .....	240
12.	Glossary.....	242

### Document Revision History

Revision	Date	Remarks regarding this Revision / Modifications
-	2004-08-19	First Draft
A	2004-09-30	Second Draft, then first official Release - Various corrections and modifications - Various additions
B	2004-12-01	Release B - Various corrections and modifications - Various additions
C	-	(skipped - not released)
D	2005-04-22	Release D - Various corrections and modifications - Various additions - Description for control functions added
E	2006-02-17	Release E - Document Revision History (this page) added - Various small corrections - Section 5.3.11 'HiTi' - reworked - Section 5.3.25 'SendHyst' - reworked - Section 5.3.26 'LimitHyst' - reworked - Section 8.2 'Scaling Groups' - reworked - Section 8.3 'Dynamic Limit Groups' - added 'Important Restriction' remark - Section 8.4 'Transmitter Types' - new - Sections 10.1 ff renamed to 11.1 ff - Section 9 split in two parts: Sections 9.7 ff renamed to 10.1 ff. - Section 9.6 'Application Watchdog for OS SW Tasks' - new - Section 9.7 'Audible Alarm & Horn Stop' - reworked - Section 9.8 ff 'Time Synchronization & Setting' - new - Section 10.8 'Loading SoftPLC7 programs' - reworked - Section 11.2 'Tips on IMAC L Screen Size / Resolution issues' - new - Section 11.3 'Tips for printing with IMAC L' - new - Section 11.4 'IMAC L Hints & Tips ('HiTi')' - new - Section 11.5 'User Rights and 'Station in Control' Concept' - new - Section 11.6 'Terminating the IMAC L tasks of an OS' - new
F	2006-04-19	Release F - Section 9.6 'FC IMAC_OS_MON_CALL' renamed. Was FC90, is now FC89 - Section 10 and all Sub-Sections Instance DBs 'DB18x' renamed to 'DB28x' - Section 10 with all its subsections: reworked acc. to the implementation of the S7-Software - Section 10.1.2.1 FC90 renamed to FC89 (FC90 was already in use for other purpose) - Section 11.7 - 'Dynamic HMI Elements for Process Control' - new
G	2009-06-26	Release G - Many corrections to meet the new AlphaVision Version 10.x - Section 2.1.3 'Use of Identification Letters' corrected to reflect IMAC L implementation - Section 5.3.39 'Smart Help' Button' inserted, all following sections renumbered (+1) - Section 5.3.41 'Type Indication NO / NC for Binary MePos' - new - Section 5.3.42 'Change Type to ...' Button for binary 'NO/NC' MePos' - new - Section 6.1 'Simatic S7 Hardware Configuration' - Time-of-Day Sync. Settings added. - Section 6.2 'Simatic S7 NetPro Configuration' - new - Section 7 'Generating ...' - reworked to reflect changes for new AlphaVision 10 - Section 8 'Engineering in AvET' - reworked to reflect changes for new AlphaVision 10 - Section 8.2.1 'EXCEL Tank Curve' Tool' - new - Section 9.7.1 'Automatic Retrieval of IMAC L Object Numbers' - new - Section 11.5.2 'ReCompile the SoftPLC Program for 'OS 99' - reworked - Section 10.8 'Converting SoftPLC7 Programs' - reworked - Section 11.8 'Ship Specific Settings' - new

### A. General References

Should further information be required or should problems arise that this description does not cover adequately enough, please contact your Siemens marine engineering partner to receive the desired information.

We would like to state, that the contents of this description does not exist due to prior arrangement, consent or a legal position and does not alter any existing agreements.

All Siemens commitments ensue from the respective sales contract, which also contains the complete and solely valid warranty control. These contractual warranty determinations will not be expanded or limited by the execution of this system description.

In addition, no claims on the efficiency of the individual customer specific units can be deflected by this description. The contractually agreed upon scope of delivery for each specific contract is the decisive factor for the exact scope of supply for each unit.

### Warning

During the operation of electrical appliances or units, it is inevitable that certain parts of these units will contain high voltage.

When warnings are disregarded, grievous bodily harm or material damage can occur.

Only qualified personnel should operate or work in proximity with these appliances or units.

These personnel must be familiar with all the warnings and maintenance measures according to this system description or related descriptions e.g. SIMATIC-PLCs.

The flawless and safe operation of these appliances and units will be provided by adequate transport, special storage, mounting and installation as well as thorough service and maintenance.

Siemens keeps the right to change technical details without prior notification.

## 1. Introduction

### 1.1 About this description

This document is designed as a guideline for the structuring and engineering of an IMAC L Integrated Monitoring and Control system. Additionally it shall help to easily perceive the structure of an installed IMAC L system and to understand abbreviations and terms used in conjunction with measuring Points (MePos).

- The first part of this document deals with basic aspects of the system structure and design of IMAC L.
- The second part describes basic topics regarding Measuring Points (MePos)
- The third part introduces the IMAC L front-end application (MS -Excel®)
- The fourth part explains the editing of MePos using the 'Edit' dialogue including a detailed description of all fields in this dialogue
- The fifth part gives some application examples for basic types of measuring points
- A list of abbreviations and acronyms (Glossary) can be found at the end of the document

It is recommended to have an introduction to the IMAC L Human Machine Interface (HMI) prior to reading this document. This document will clarify all basic IMAC L properties, features and the terms used in conjunction with IMAC L. Please refer to the document 'IMAC 55 Human Machine Interface (HMI) Description' (document number E10231-Y1009-U010).

### 1.2 Symbols and conventions

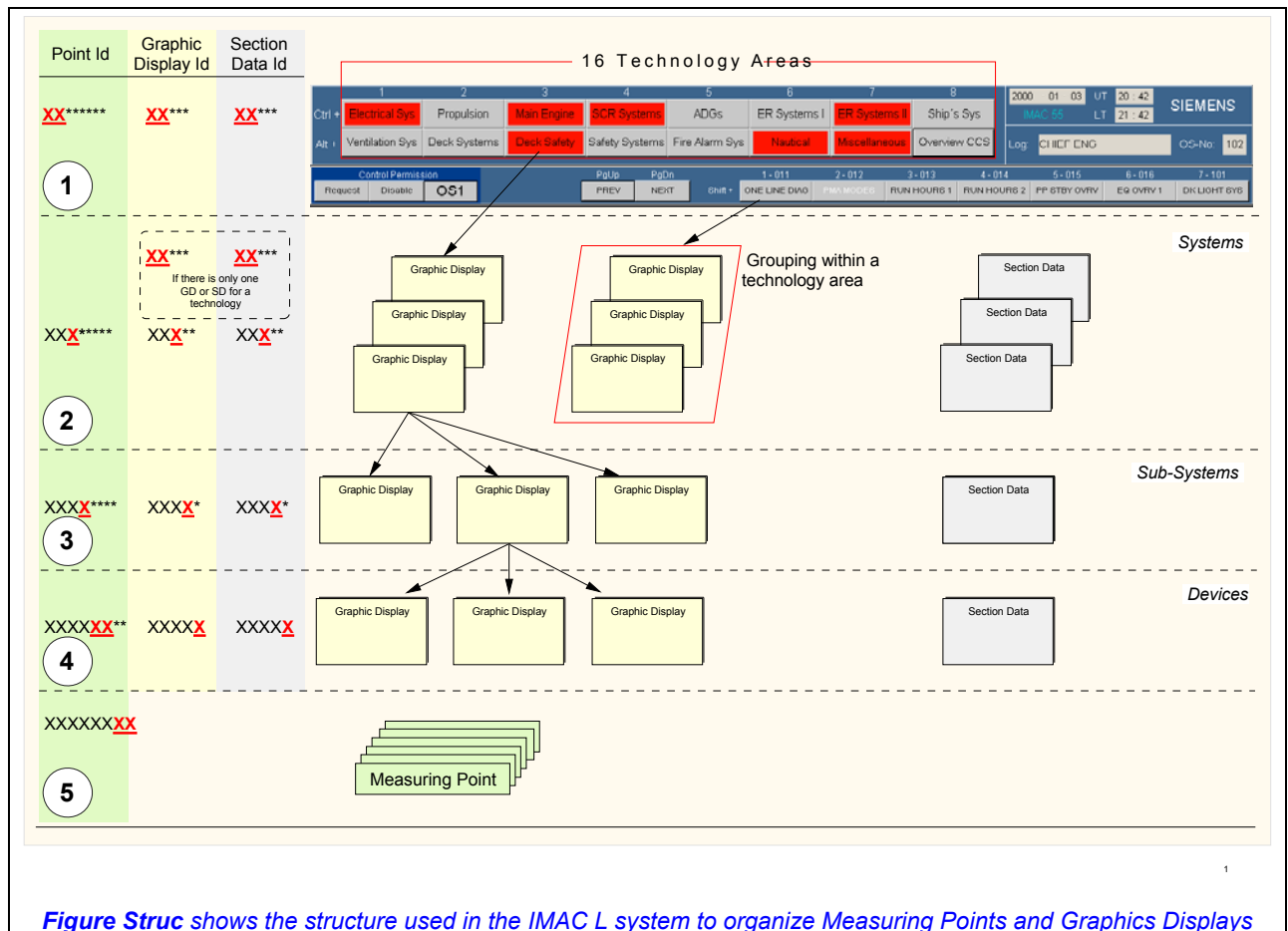
This description utilises the following conventions and symbols:

- Names and characterisations of the IMAC 55 components are written in small capitals, e.g. ALARM SUMMARY, EVENT LOG etc.
- References to other pages within this description will be written in straight brackets, e.g. [→ Log in, page #]
- References to graphics depicted in this description will be made in the form of "→ Figure. "... on page ##".
- Several abbreviations will be used in so called "Hungary Notation" : e.g. AAGrp for **A**udible **A**larm **G**roup. This is a writing style that is common for the use in conjunction with computer systems.

## 2. Basic Structure of IMAC L

An IMAC L system can possibly comprise several thousand Measuring Points (MePos) and a large number of process graphics displays (GDs).

In order to easily manage this information flood, IMAC L provides several logical structuring or hierarchy levels.



**Figure Struc** shows the structure used in the IMAC L system to organize Measuring Points and Graphics Displays

### 2.1 Technology Areas, SECTION DATA and Process Graphics Displays

The highest level ① is used to differentiate between the 'Technology Areas' of a ship. Examples for technology areas are 'Main Propulsion System', 'Cooling Water Systems' or 'Electrical Systems'. IMAC L supports up to 16 technology areas. One of these 16 areas is exclusively reserved for IMAC L system MePos and displays. The other areas have to be defined as part of the system design.

The technology areas will be displayed as part of the header on every IMAC L graphics display. Every technology area has a button assigned to it, which will navigate directly to a GD or SECTION DATA display in this area that is used as an entry point to monitor and control the respective area.

SECTION DATA displays are automatically generated lists of measuring points belonging to a specific functional group on a ship. The assignment of every MePo to a section is part of IMAC L MePo engineering.

If an alarm is active in a technology area, the appropriate header button will be flashing red, if there is at least one unacknowledged alarm in this area. If all alarms in one area are acknowledged, the corresponding button will be steadily red instead. Optionally IMAC L can offer a header area that will display the active number of alarms in every technology area button. For this design option, only 8 technology areas are possible instead of 16.

The next structuring level below 'Technology Areas' ① is called 'Systems' ②. This level contains one main process graphics display (GD) or SECTION DATA list for each technology area. If this main process level requires a further breakdown, the next levels 'Sub-Systems' ③ and 'Devices' ④ are available for convenient structuring.

These levels are logical levels that are usually determined according to technological requirements. A systematic identification and naming scheme for MePos, GDs and SECTION DATA lists as depicted in Figure 1 will be useful for an easy operator access to this structure.

Example:

Technology Area	①:	Ships Electrical Systems
1 <sup>st</sup> Sub-Level 'Systems'	②:	Generator Sets and + Power Distribution
2 <sup>nd</sup> Sub-Level 'Sub-Systems'	③:	One single Generator Set
3 <sup>rd</sup> Sub-Level 'Devices'	④:	The single devices of this Generator Set

If Sub-Levels ③ and ④ are used in a technology area, the MENU AREA (below the header area) will contain so called 'ROUTING BUTTONS', that will allow to easily navigate between all GDs of this technology area according to the logical structure of this area.

### 2.1.1 Structuring, Identification System

A logical and meaningful identification of measuring Points (MePos), SECTION DATA and Graphics Displays (GDs) will significantly ease the operation of a control and monitoring system.

The scheme as shown below is meant to be an example. Specific project requirements may be a cause to use alternative criteria for the design of a system structure.

### 2.1.2 Structuring, Identification for Measuring Points

**Hint:** IMAC L does not distinguish between the use of upper case and lower case letters for the naming of MePos and GDs.

For the structuring of MePo PointIDs it is recommended to use the same scheme as it is used for SECTION DATA lists and GDs. Usually there are significantly more MePos than SECTION DATA lists or GDs. That is, why a MePo PointID requires more characters for unique identification.

**Technology Area** / "Main Level"  
①: **XX+++++** (the first two characters, Range 01 - 16)

Examples:

- 01 Electrical System
- 02 Propulsion
- 03 Main Engine
- 04 Auxiliary Generators
- 05 ER Systems
- 12 Ballast System
- 16 IMAC L System

Alternatively characters can be used instead of numbers to identify a technology area. Nevertheless, finding meaningful abbreviations using only two characters has its limitations. Using a straight 'numbering scheme' is usually easier and better.

**System Level** / 1st Sub-Level  
②: **++X++++** (the third character, Range 0 - 9)

In the System level the third character will identify the System within a technology area.

Examples:

- 1 Auxiliary Generator Set 1
- 2 Auxiliary Generator Set 2
- 3 Auxiliary Generator Set 3
- 4 Auxiliary Generator Set 4
- 5 Emergency Generator Set
- 6 Shore Connection
- 7 Power Distribution

**Sub-System Level** / 2nd Sub-Level  
③: **+++X++++** (the fourth character, Range 0 - 9, A - Z)

Examples:

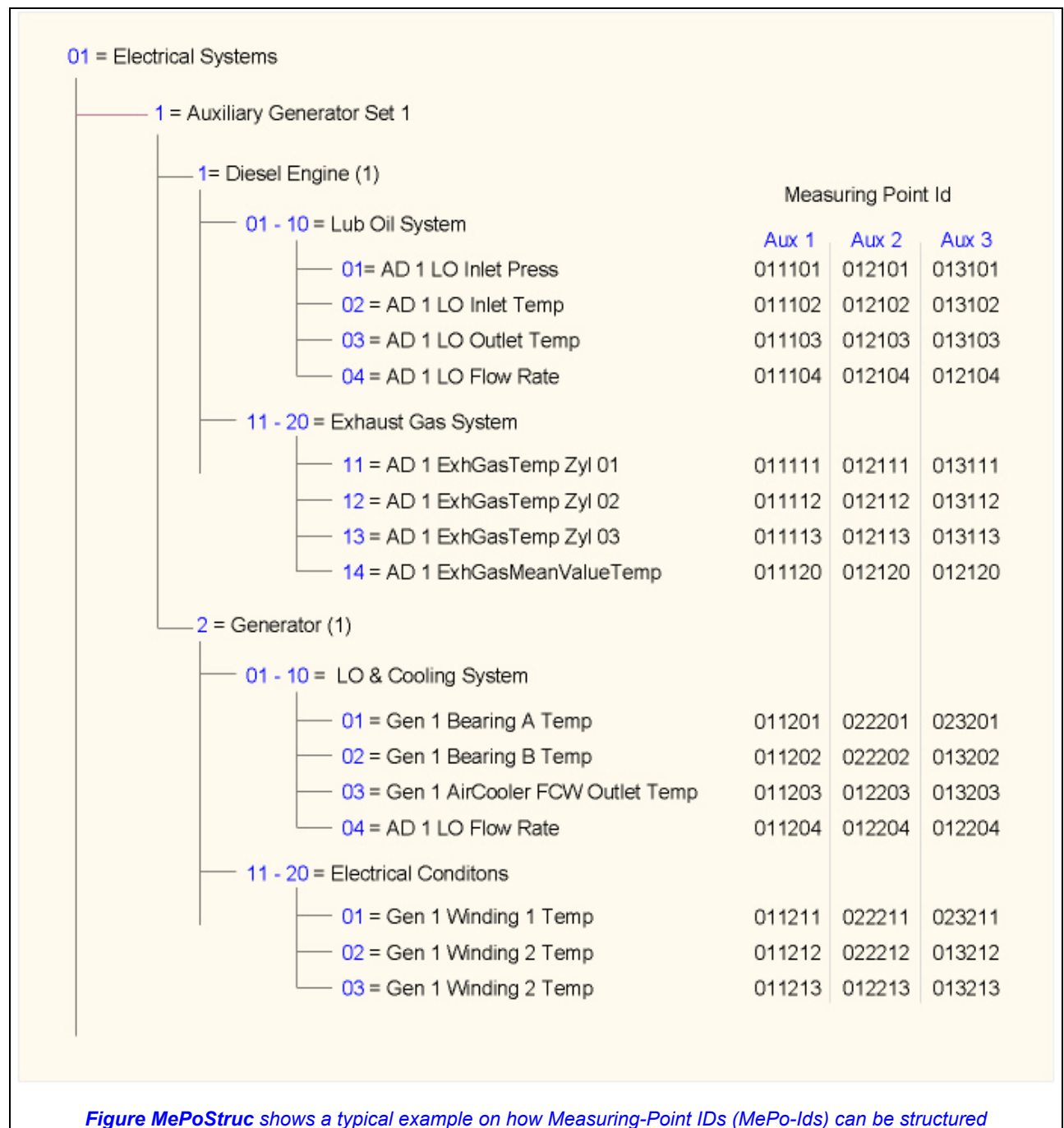
- 1 Diesel Engine 1
- 2 Generator 1

**Devices Level** / 3rd Sub-Level  
④: **++++XX++** (the fifth & sixth character, Range 0 - 9, A - Z)

Examples:

- 01 - 10 DE 1 Lub Oil System
- 11 - 20 DE 1 Exhaust Gas System





### 2.1.3 Use of Identification Letters in MePo-Ids to identify internal Measuring Points

Various measuring points will require one or more related IMAC L-points representing e.g. internal results of calculations or other evaluation results. In order to be able to assign these additional points to the (hardware) MePo they are originating from, the MePo-Id of the originating point will be retained and an identification letter will be added to specify the purpose of this internal MePo.



Identification Letter	Measuring Point Type	Origin / Unit	Explanation
<b>Generic</b>			
I	INTERNAL MEASURING POINT		Will be used, if an existing HW-MePo requires an internal MePo (e.g. for measured value after linearization).
R	RUNNING HOUR COUNTER	(internal)	MePo for counting running hours of a device
Y	Dummy	(internal)	For use in situations requiring a dummy MePo.
<b>Exhaust Gas Mean Value Calculations</b>			
D	DEVIATION		Used for computing Exhaust Gas Temperature Mean Value Deviations of Diesel Engine Cylinders.
M	MEAN VALUE		Used for computing Exhaust Gas Temperature Mean Values of Diesel Engine Cylinders.
O	OFFSET, Offset to Mean Value		Used for the temperature offset value of each single cylinder compared to the exhaust gas mean value after symmetrization.
<b>Control Functions (e.g. Valves, Pumps)</b>			
(-)	Alarm Status Word	(internal)	Represents the current alarm condition of a device to be controlled (for text indication of alarm cause)
A	Alarm Flag	(internal)	Represents the fault condition (alarm active / inactive) of a device to be controlled.
C	Control Word		Word containing the control input from the HMI (OS) to a device.
S	Status Word	(internal)	Used to control the graphical representation (e.g. figure colours & shape) of a device's process condition. The values of 'status' correspond to a figure condition table in the HMI system. (Analogue MePos only).
B	Stand By-marker	(internal)	Used to indicate the 'Stand By' condition of a device.
<b>For Tank Contents Measurement &amp; Calculation</b>			
(none)	Pressure	[bar]	This MePo will be used by IMAC L to measure the hydrostatic pressure in a tank. Usually this is a hardware MePo with a 4-20mA type of signal.
D	Media Density $\rho$	(internal) [tons/m <sup>3</sup> ]	Internal MePo representing the density of the medium presently stored in a tank.
L	Filling Level 'h'	(internal) [m]	This MePo receives the tank contents (filling level) calculated from the tanks pressure MePo. The temperature compensated media density will be used for this calculation. If a filling level offset for the pressure sensor's mounting position has been assigned, this will be taken into account for the calculation as well.
M	Mass ,m'	(internal) [tons]	This MePo shows the tank contents (mass) calculated by IMAC L based on "Volume x Media density".
R	Reference Temp. $T_1$	(internal) [°C]	Reference temperature for the specified specific media density. Required, if temperature compensation of media density is necessary.
S	Scale Group	(internal)	This MePo will be used by IMAC L to reference the scaling Group used to calculate the Tank Form curve
T	Tank Temp. $T_2$	[°C]	This MePo will be used by IMAC L to measure the media temperature of a tank. Required, if a ONLINE temperature compensation of media density is necessary.
V	Filling Volume ,V'	(internal) [m <sup>3</sup> ]	This MePo shows the tank contents (volume) calculated by IMAC L based on the tank form curves.
X	Volumetric Expansion Coefficient $\beta$	(internal) [1/K]	Volumetric Expansion Coefficient used for temperature dependent media density compensation. Required, if temperature compensation of media density is necessary.
C	Tank Temp. $T_2$	(internal) [°C]	This MePo will be used by IMAC L to manually input the media temperature of a tank. Required, if a OFFLINE temperature compensation of media density is necessary.

*Table IdentLetters shows identification letters used to distinguish additional MePos required by IMAC L^^*

### IMAC L Standard Documentation

### Engineering Guideline Basic Structure of IMAC L

Point Id	Description	Status 1	Status 2	Status 3	Status 4	Status 5	Status 6	Status 7	Status 8	Remark
Example: Ballast system valve no. 35										
121V35	BalSys Valve 35 Alrm.Stat-Word	Normal	Ext. Fail	Moving Al.	Open. Al.	Clos. Al	No FD-Back	Locked	Emcy Off	Alarm Stat-Word
121V35A	BalSys Valve 35 Alrm.-Status	Normal	Failure							Alarm Flag
121V35C	BalSys Valve 35 OS Cmd-Word	SysIntern								OS Cmd Word
121V35S	BalSys Valve 35 OS Status-Word	SysIntern								OS Status word
Example: Tank contents measurement of ballast water tank no. 01										
123T01	WB No 01 Deep Tank	Display								Pressure MePo
123T01D	WB No 01 Deep Tank	Display								Density
123T01L	WB No 01 Deep Tank	Display								Filling Level
123T01M	WB No 01 Deep Tank	Display								Filling Mass
123T01R	WB No 01 Deep Tank	Display								Reference temp.
123T01S	WB No 01 Deep Tank	SysIntern								Scal.Group
123T01T	WB No 01 Deep Tank	Display								On Line Temp.
123T01V	WB No 01 Deep Tank	Normal	MaxLevel	Overflow						Filling Volume
123T01X	WB No 01 Deep Tank	Display								Volum.Exp.Coeff.
Example : Exhaust gas temperatures of main engine no. 01 (partially)										
01101	ME ExhaustGasTemp Cyl 1	Normal	Too High	Slow Down						Exh. Gas temp.
01101D	ME ExhaustGasTemp Cyl 1 Dev	MaxNeg	Normal	MaxPos						Exh. Gas deviat.
01101O	ME ExhaustGas Cyl 1 SymOffset	Display								
01102	ME ExhaustGasTemp Cyl 2	Normal	Too High	Slow Down						Exh. Gas temp.
0110M	ME ExhaustGasTemp Mean Value	Display								Exh. Gas Mean T
Example: Ballast water pump no. 01										
12P10	Ballast Pump 1 Alrm.Stat-Word	Normal	Ext. Fail	Auto-Fail	Action-Al.	No FD-Back	Locked	Emcy Off	BlackOut	Alarm Stat-Word
12P10A	Ballast Pump 1 Alrm.-Status	Normal	Failure							Alarm Flag
12P10C	Ballast Pump 1 OS Cmd-Word	SysIntern								OS Cmd Word
12P10S	Ballast Pump 1 OS Status-Word	SysIntern								OS Status word
12P10R	Ballast Pump 1 Running Hour	Running	Stopped							RunningHour

*Table IdentLettersExample shows an example MePo list with additional points required by IMAC L (distinguished by identification letters appended to the MePo name)*

### 2.1.4 Structuring, Identification for Process Graphics Displays (GDs)

Process graphics displays (GDs) can be named according to the scheme used for measuring points (MePos) as well. Nevertheless, a fully detailed structure is only required for applications with extremely many GDs. Simple configurations with only a few GDs may use a simplified naming scheme for GDs. Nevertheless even here the 'root' of the naming scheme based on the technology groups & systems should be preserved:

Example for Propulsion System (in technology group 2) :

Main Engine 1: GD-Names = 0201xx (xx = consecutive numbering)  
Main Engine 2: GD-Names = 0202xx

### 2.1.5 Structuring, Identification for Section Data

Section Data identifiers will usually be assigned to Systems (level ② of the basic structure. Every measuring point in IMAC L has to be assigned to a technology area and a section.

## 3 Measuring Points Lists - Basics

### 3.1 Measuring Point Description Texts

Measuring point description texts are meant to describe a single MePos subject, but not it's current condition. An example for a MePo description text is "ME1 NOZZLE COOLING OIL PP".

The condition of a MePo will be described in the 'Status Text' fields of this MePo. For the example above the conditions of this cooling oil pump could for example be 'NORMAL' and 'FAULT'.

One more example on how to correctly separate 'Subject' and 'Status' of a MePo :

Description	Status 1	Status 2	
DG2 Bearing Temp AFT	Normal	High	OK
DG2 Bearing Temp AFT too high	Normal	Alarm	WRONG !

Within a MePo description text the Subject itself (e.g. the main device or aggregate) or the most essential description should be mentioned first (i.e. hierarchically descending order of 'importance' within the text) :

Description	
ME1 Cyl 1 Exhaust Gas Temp	OK
Cyl 1 Exhaust Gas Temp ME 1	WRONG !

Identical Objects shall have identical names throughout the system. It's a safe method to confuse any user if you have a variety of names for identical objects (e.g. Diesel Generator, Generator, Auxiliary Generator or Auxiliary Engine are finally all the same thing). The term that is commonly used on board (customers choice) shall be solely used.

#### 3.1.1 Engineering Guideline

The limited number of characters available for MePo description and status texts makes it necessary to use abbreviations. In order to provide a user-friendly design of IMAC L the following rules are introduced :

- Identical types of Objects, functions (or equivalent) shall always be abbreviated in the same way (i.e. it is not allowed to refer to one Pump as 'PP' and to another as 'PMP').
- If an abbreviation has been introduced to a project, all references in this IMAC L description and status texts shall use this abbreviation only. The use of the full term (non-abbreviated) is not allowed any more.

Example: If for a project the abbreviation list says, that a pump has to be referenced to as 'PP', it shall not be allowed to use the term 'Pump' or 'PUMP', even if for single measuring points there would be enough text space available to use the full term.

- Abbreviations in Texts will use the so-called 'Hungarian notation' to improve readability and to save text space. Examples : HydrPp (Hydraulic Pump), SettITk (Settling Tank).
- Stand-alone Abbreviations like "PMS" (for Power Management System) should always be written in capital letters only.
- If the 30 characters for a MePo description are not sufficient, it is optionally possible to use the 'Hints and Tips' (HiTi) texts for a more detailed description.

**Hint:** It is strongly recommended to introduce mandatory abbreviations lists for every project. The lists as shown in the next paragraphs may serve as a basis for project specific tailoring. These lists should be discussed and agreed upon with the customer prior to any MePo engineering ! This avoids undesired inconsistencies in MePo naming and thus potentially saves lots of efforts !

### 3.1.2 Basic Abbreviations List (English / German) for Measuring Point Description Texts

Abbrev.	Description	Abkürzung	Erläuterung
		Vent	Ventilator
		Zul	Zulüfter
#	After this character a hint on system internal relation, e.g. blocking source or a relating Point Id, will follow.	#	Nach diesem Zeichen folgt ein Hinweis auf systeminterne Zusammenhänge. z.B. Blockierungen oder eine Referenz-Meßstelle.
+	This character at the end of the description field means, there are additional information (Hints and Tips)	+	Dieses Zeichen am Ende des Feldes verweist auf weitere hinterlegte Informationen (HiTi).
AC	Air Condition	KA	Klimaanlage
AC	Alternating Current	WS	Wechselstrom
AD	Auxiliary Diesel	HiDi	Hilfsdiesel
AS	A-Side, e.g. for Engine	AS	A-Seite, z.B. beim Motor
Bal	Ballast System	Bal	Ballastsystem
BS	B-Side, e.g. for Engine	BS	B-Seite, z.B. beim Motor
BTr	Bow Thruster	BsR	Bugstrahlruder
CMSB	Consumer Main Switch Board	BnT	Bordnetztafel
CPP	Controllable Pitch Propeller	VSP	Verstellpropeller
CRP	Contra Rotating Propeller	CRP	Gegenläufig rotierende Propeller
CTF	Consumer Transformer	BnTo	Bordnetztransformator
CW	Cooling Water	Kw	Kühlwasser
CWS	Chilled Water Set	KwS	Kaltwasser-Satz
Cyl	Cylinder	Zyl	Zylinder
DB	Double Bottom (Tank)	DB	Doppelboden (-Tank)
DC	Direct Current	GS	Gleichstrom
DG	Diesel Generator	DG	Dieselgenerator
DFO	Diesel Fuel Oil	DO	Dieseloel
Drn	Drain	Abf	Abfluss
E	East	O	Ost
EAS	Extended Alarm System	EAS	Wachalarm-System
ECR	Engine Control Room	MKR	Maschinenkontrollraum

Abbrev.	Description	Abkürzung	Erläuterung
EFan	Exhaust Fan	AbL	Ablüfter
ER	Engine Room	MR	Maschinenraum
ESB	Emergency Switch Board	NoTa	Notschalttafel
ExG	Exhaust Gas Temperature	AbG	Abgastemperatur
Fan	Fan	Lft	Lüfter
FCW	Fresh Cooling Water	FkW	Frischkühlwasser
Fli	Flowing in...	Zuf	Zufluss
Flt	Filter	Flt	Filter
FO	Fuel	Bst	Brennstoff
HFO	Heavy Fuel Oil	SwO	Schweröel
HT	High Temperature (Coolant)	HT	Hoch-Temperatur (Kühlkreislauf)
I>>	Schort Circuit	I>>	Kurzschluß
Lev	Level	Niv	Niveau
LO	Lubrication Oil	SO	Schmieröel
LT	Low Temperature (Coolant)	NT	Niedrig-Temperatur (Kühlkreislauf)
MCC	Motor Control Center	MCC	
ME	Main Engine	HM	Hauptmotor
MER	Main Engine Room	HmR	Hauptmaschinen-Raum
MSB	Main Switch Board	HaTa	Hauptschalttafel
MvD	Mean Value Deviation (Exhaust)	MwA	Mittelwertabweichung (Abgas)
N	North	N	Nord
NE	North/East	NO	Nord/Ost
P	Port (side)	Bb	Backbord
PCM	Pump Control and Monitoring	PCM	Pumpensteuerungs-Automatik
PCS	Propulsion Control System	PCS	Fahreranlage-Überwachungssys
PCU	Process Control Unit	PCU	SIMATIC-Unterstation
PH	Pre Heater	VE	Vorerhitzer
PLC	Programable Logical Contoller (e.g. SIMATIC S5/S7)	SPS	Speicher-Programmierbare Steue- rung (z.B. SIMATIC S5/S7)
PMS	Power Management System	EEA	Energieerzeugerautomatik
Pp	Pump	Pp	Pumpe
Pr	Pressure	Drk	Druck
PS	Power Supply	SV	Stromversorgung
PSys	Propulsion System	Fant	Fahrtrieb
RhC	Running Hour Counter	BsZ	Betriebsstundenzähler

Abbrev.	Description	Abkürzung	Erläuterung
S	South	S	Süd
SaSt	Safety Station	SiSt	Sicherheitsstation
SCW	Sea Cooling Water	Skw	Seekühlwasser
SeC	Sea Chest	SKt	See-Kasten
STB	Starboard (side)	STB	Steuerbord
SW	South/West	SW	Süd/West
Sys	System	Sys	System
TC	(Exhaust Gas) Turbo Charger	ATL	Abgas-Turbolader
Temp	Temperature	Temp	Temperatur
Tk	Tank	Tk	Tank
Va	Valve	Vt	Ventil
W	West	W	West

### 3.1.3 Basic Abbreviations List (English / German) for Measuring Point Status Texts

Abbrev.	Description	Abkürzung	Erläuterung
ActivNack	ACTIVE AND NOT ACKNOWLEDGED, i.e. used for a Visual Alarm Group.	ActivNack	Aktiv und nicht quittiert. Status für z.B eine Visual Alarm Group.
AutoStop	Status indicating an automatic stop of a device caused by operational parameters being out of their permissible range.	AutoStop	Meldung für die Auslösung eines automatischen Stop für ein Aggregat aufgrund unzulässiger Betriebsparameter.
BlackOut	Loss of power supply. Used mainly in conjunction with pump or valve control. Control has detected a black out and initiates appropriate actions to remedy this situation after return of power supply (i.e. sequenced starting).	BlackOut	Netzausfall. Im Zusammenhang mit der Steuerung von Pumpen, Lüftern oder anderen Aggregaten: Die Steuerung hat einen Black Out erkannt und leitet ggf. hieraus ein bestimmtes Verhalten ab; z.B. gestaffelten Wiederanlauf oder ähnliches.
ComFail	Communications failure. The communication link on a serial interface or bus connection is faulty	KommFehl	Kommunikations-Fehler: Die serielle Kopplung zwischen Geräten oder eine Busverbindung ist gestört.
ContrFail ActnAlarm	Control failure (or 'Action Alarm'). A control command has been sent to a device, but the device does not react appropriately. Possible cause is for example a missing power supply from consumer switchboard for this device.	SteuerFehl	Steuerungs-Fehler. An eine Anlage oder ein Gerät wurde ein Steuer-Kommando gesendet, aber die Anlage oder das Gerät reagieren nicht; z.B. weil die Versorgungsspannung am Schaltgerät fehlt.
Display	This MePo serves for indication purposes only.	Anzeige	Diese Meßstelle dient ausschließlich zur Anzeige von Meßwerten.



Abbrev.	Description	Abkürzung	Erläuterung
Dummy	For internal purposes only.	Dummy	Nur für interne Zwecke.
ExtFailure	Used mainly in conjunction with pump or valve control. These control systems can receive a 'failure' message from a 'higher level' system (i.e. stand-by automatic system) and will indicate this status as a consequence. Some control functions will use this status as a standard, even if in a specific application no 'higher level' system is present.	ExtFehler	Eine Pumpensteuerung oder eine Ventilsteuerung oder ein anderes Aggregat kann durch eine "übergeordnete" Automatik eine Fehlermeldung erhalten. Beispielsweise kann bei einer Pumpensteuerung eine vorgeschaltete Stand By-Automatik diese Fehlermeldung auslösen. Dieser Statustext ist bei einigen Steuerungen "standardmäßig" vorhanden, auch wenn keine externe Fehlermeldung vorgesehen ist.
Failure	The device or aggregate has a failure	Gestoert	Das Aggregat oder Gerät ist gestört.
FBackFail	Feed Back Failure. A valve or flap is monitored for it's limit switch positions. If IMAC L detects, that a feedback signal that previously has been correctly detected is now missing, without any control commands via IMAC L causing this, this alarm status will be issued. Possible causes are manual interference at device level, sensor failures (limit switch broken) or wire break of sensor cable.	RM-Fehl	Rückmeldungs-Fehler. Bei einem Ventil oder einer Klappe werden die Endstellungen überwacht (z.B. mittels Endlagenschalter). Erkennt IMAC L nun, dass die Rückmeldung plötzlich nicht mehr vorhanden ist, ohne dass zuvor ein Steuerbefehl abgesetzt wurde, so wird der Alarm ausgelöst. Mögliche Ursachen könnten eine Verstellung von Hand sein, oder aber auch ein Sensor- (Endlagenschalter) Fehler.
IllegalVal	Illegal value. Used in conjunction with measuring points, that report a limit infringement using a <u>binary contact</u> for detection (exceeding high-limits, under-running low-limits).	Unzul Wert	Unzulässiger Wert. Hierbei handelt es sich in der Regel um Meßstellen, die über <u>einen binären Kontakt</u> eine Grenzwertüberschreitung in positiver oder negativer Richtung melden.
Inhibiting	If this status is entered, the interlocking (BLOCKING) of one or more other MePos will be initiated. If any of the blocked MePo has a limit infringement, this will now be ignored (no alarm). [→ status text: "Releasing"] [→ Document 'IMAC L HMI Description', Keywords "Inhibit Summary, Blocked Summary".].	Sperrend	Beim Eintritt in diesen Status wird die Alarmauswertung einer anderen Meßstelle unterdrückt. (Inhibited). Das Sperren kann auch gleichzeitig mehrere anderen Meßstellen betreffen. [→ IMAC L-Systembeschreibung, Stichworte "Inhibit Summary, Blocked Summary".]
Locked	The operation of a device or a plant has been locked by the control system (i.e. caused by a 'repair switch').	Gesperrt	Die Betätigung eines Gerätes oder einer Anlage über das Prozeßleit-System wurde gesperrt. z.B. mittels eines "Reparaturschalters".



Abbrev.	Description	Abkürzung	Erläuterung
MaxLevel	The maximum filling level of a tank, bilge or similar has been exceeded.	MaxFuellst	Der maximal zulässige Füllstand eines Tanks, einer Bilge usw. ist überschritten.
MaxNeg	Maximum value for negative deviations has been exceeded (i.e. exhaust gas mean value deviation).	MaxNeg	Maximal-Wert in negativer Richtung; z.B. Abgas-Mittelwert-Abweichung.
MaxPos	Maximum value for positive deviations has been exceeded (i.e. exhaust gas mean value deviation).	MaxPos	Maximal-Wert in positiver Richtung; z.B. Abgas-Mittelwert-Abweichung.
MinLevel	The minimum filling level of a tank, bilge or similar has been under-run.	MinFuellst	Der minimal zulässige Füllstand eines Tanks, einer Bilge usw. ist unterschritten.
MTEceed cl	Monitoring time for a valve has exceeded its limit (Moving time 'to close', closing alarm). A valve has been ordered to move from its opened to its closed position, but does not reach its desired position within a defined monitoring time. Examples for possible causes are low hydraulic oil pressure or a mechanical jam of the valve.	LZFehl zu	Laufzeitüberwachung für Ventile. Ein Ventil soll per Steuerung von der Auf- in die Zu-Stellung gefahren werden. Das Ventil erreicht jedoch die Zu-Stellung nicht innerhalb der eingestellten Überwachungszeit; z.B. weil der Servoöldruck zu niedrig ist oder das Ventil klemmt.
MTEceed op	Monitoring time for a valve has exceeded its limit (Moving time 'to open', opening alarm). A valve has been ordered to move from its closed to its opened position, but does not reach its desired position within a defined monitoring time. Examples for possible causes are low hydraulic oil pressure or a mechanical jam of the valve.	LZFehl auf	Laufzeitüberwachung für Ventile. Ein Ventil soll per Steuerung von der Zu- in die Auf-Stellung gefahren werden. Das Ventil erreicht jedoch die Auf-Stellung nicht innerhalb der eingestellten Überwachungszeit; z.B. weil der Servoöldruck zu niedrig ist oder das Ventil klemmt.
NAct NAckn	Not active and not acknowledged. Status used i.e. for IMAC L VISUAL ALARM GROUPS.	NAct NAckn	Nicht aktiv und nicht quittiert. Status für z.B. eine VISUAL ALARM GROUP.
Not Active	Not active. Mainly used for system internal purposes (i.e. to indicate, if a VISUAL ALARM GROUPS is active).	Not Active	Nicht aktiv. Benutzung hauptsächlich für systeminterne Meldungen; z.B. ob eine Visual Alarm Group aktiv ist.

Abbrev.	Description	Abkürzung	Erläuterung
p<StBy-Stp	Pressure is below the 'switch off' pressure limit for a stand-by pump. Example: a gear has a lubrication oil pump driven by the rotation of the gear itself. If the gear's lube oil pressure will fall below a minimum limit, an electrically driven stand-by pump has to cut in to ensure sufficient oil supply to the gear. This status describes the situation that the pressure has fallen below its defined limit. [→ StBy-Stopp, StBy-Anf]	p<StBy-Stp	Druck unterhalb des "Abschalt-Druckes" für eine Stand By-Pumpe. Ein Aggregat, z.B. ein Getriebe verfügt über eine "angehängte" Schmierölpumpe. Unterhalb einer bestimmten Drehzahl bzw. eines bestimmten Schmieröldruckes wird eine elektrisch angetriebene StBy-Pumpe betrieben. [→ StBy-Stopp, StBy-Anf]
p>StBy-Req	Pressure is above the 'switch off' pressure limit for a stand-by pump. Example: a gear has a lubrication oil pump driven by the rotation of the gear itself. If the gear's lube oil pressure will fall below a minimum limit, an electrically driven stand-by pump has to cut in to ensure sufficient oil supply to the gear. This status describes the situation that the pressure is now above its defined limit. [→ StBy-Stopp, StBy-Anf]	p>StBy-Anf	Anforderung einer Stand By-Pumpe zurücknehmen. Ein Aggregat, z.B. ein Getriebe verfügt über eine "angehängte" Schmierölpumpe. Unterhalb einer bestimmten Drehzahl bzw. eines bestimmten Schmieröldruckes wird eine elektrisch angetriebene StBy-Pumpe angefordert bzw. gestartet. Sobald der Druck jedoch einen Grenzwert überschreitet, wird die StBy-Anforderung wieder zurück genommen. [→ StBy-Anf, StBy-Stopp]
QuickClose	Quick closing has been initiated for a device (i.e. fire flaps in ventilation system have been actuated during fire fighting to prevent smoke propagation).	SchnellSchl	Schnellschluß. Für dieses Gerät wurde ein Schnellschluß ausgelöst. z.B. von der Feuer-/Schnellschlußstation.
Releasing	If this status is entered, the interlocking (BLOCKING) of another MePo will be released. If the other MePo has a limit infringement, this will now cause an alarm. [→ status text: "Inhibiting"].	Freigebend	Beim Eintritt in diesen Status wird die Sperrung (Blockierung) der Alarmauswertung einer anderen Meßstelle aufgehoben; d.h. diese Meßstelle gibt im Falle einer Grenzwertüberschreitung Alarm. [→ STATUS-Text: "Sperrend".]
ShutDown	Stopping of an aggregate has been initiated due to a <u>limit infringement</u> (i.e. temperature)	ShutDown	Für ein Aggregat wurde ein automatischer Stopp aufgrund eines zu hohen Wertes, z.B. Temperatur, ausgelöst.
SlowDown	Reducing of an aggregate has been initiated due to a <u>limit infringement</u> (i.e. reduction of main propulsion engine caused by one cylinders exhaust gas temperature violating its deviation limit from mean value).	SlowDown	Es wird eine Reduzierung eines Aggregates, z.B. der Hauptmaschine aufgrund eines zu <u>hohen Wertes</u> , z.B. der Abgas-Mittelwertabweichungs-Temperatur, ausgelöst.

Abbrev.	Description	Abkürzung	Erläuterung
StBy-Req	Request stand-by pump to run. Example: a gear has a lubrication oil pump driven by the rotation of the gear itself. If the gear's speed will fall below a minimum limit, an electrically driven stand-by pump has to cut in to ensure sufficient oil supply to the gear. This status describes the situation that requires the stand-by pump to run because the speed is below its defined limit. [→ StBy-Stopp, StBy-Anf]	StBy-Anf	Anforderung einer Stand By-Pumpe. Ein Aggregat, z.B. ein Getriebe verfügt über eine "angehängte" Schmierölpumpe. Unterhalb einer bestimmten Drehzahl bzw. eines bestimmten Schmieröldruckes wird eine elektrisch angetriebene StBy-Pumpe angefordert bzw. gestartet. [→ p> StBy-Anf]
StBy-Start	Stand-by start of an aggregate. Example: a pump has been switched from 'stand-by' to 'run', because the automatic pump control has detected a pressure drop.	StBy-Start	Stand By-Start. Ein Aggregat, z.B. eine Pumpe, ist aus der Betriebsart "Stand By" in Betrieb gegangen, weil z.B. von der Pumpensteuerautomatik ein Druckabfall erkannt wurde.
StBy-Stop	Switching off of a stand-by pump. Example: A gear has a lubrication oil pump driven by the rotation of the gear itself. If the gear's speed or lube oil pressure will rise above a maximum limit, an electrically driven stand-by pump can be switched off because the directly driven pump will ensure sufficient lube oil supply to the gear. This status describes the situation that requires no stand-by pump to run because the speed / pressure is above its defined limit. [→ StBy-Anf].	StBy-Stop	Abschalten einer Stand By-Pumpe. Ein Aggregat, z.B. ein Getriebe verfügt über eine "angehängte" Schmierölpumpe. Überhalb einer bestimmten Drehzahl bzw. eines bestimmten Schmieröldruckes wird eine elektrisch angetriebene StBy-Pumpe abgeschaltet. [→ p< StBy-Stp]
SystemIn	This status is required for internal MePos, which are used for controlling the appearance of HMI figures. The numerical value contained in MePos with this status will contain information to control the appearance (shape, colour) of figures via 'figure condition tables'.	SystemIn	Hierbei handelt es sich um Systeminterne Meßstellen, die für die Visualisierung von Figuren auf dem Bildschirm benötigt werden. Der angezeigte Analogwert wird dazu benutzt, um die verschiedenen Zustände einer Figur zu unterscheiden.
TestPos	Test position (e.g. for a power circuit breaker).	TestPos	Testposition für z.B. einen Leistungsschalter

## 4. IMAC L Measuring Points Lists Front-end

IMAC L will use a Microsoft Excel® spreadsheet for editing the required measuring points list. This spreadsheet will contain one row for each MePo and 142 columns containing all possible parameters for every MePo. In theory all information required for a MePo can be directly entered to the spreadsheet cells. Nevertheless it is **strongly recommended to use only the 'Edit' form** included in this spreadsheet. This Edit form will display all parameters of a MePo in a convenient and ergonomic view. Online parameter checking will automatically indicate basic faults in MePo parameters by colour changes of the appropriate field(s) :

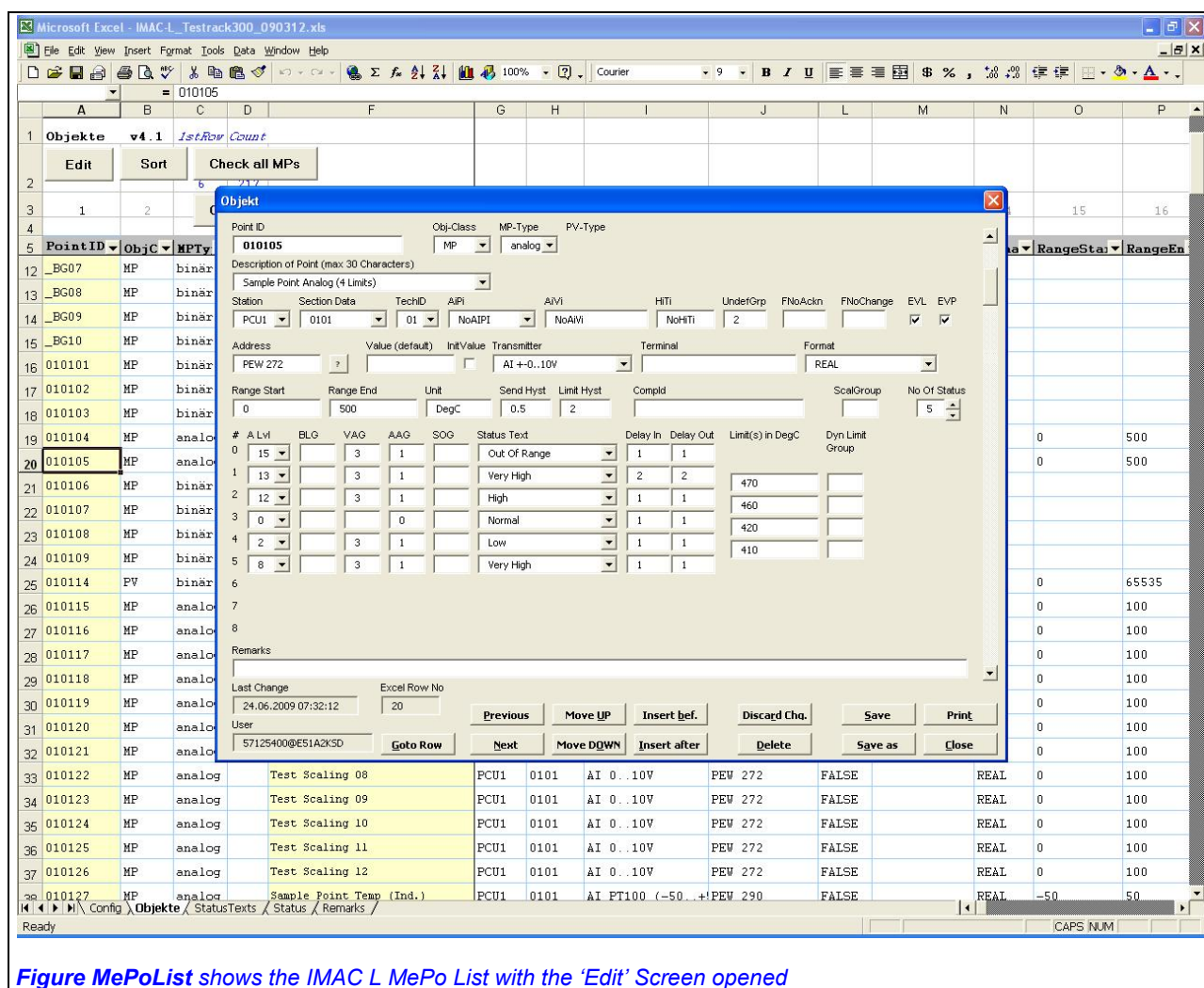


Figure MePoList shows the IMAC L MePo List with the 'Edit' Screen opened

Depending on the type of the MePo only those fields required for this type of point will be presented to the user.

This Excel spreadsheet will be referred to as the IMAC L 'MePo front-end'.

## 4.1 Opening the Excel MePo Front-end

Open the MS-Excel file with Microsoft Excel (Excel 2000 or higher). The following dialogue will ask, if the use of Macros shall be permitted :



Select '**Enable Macros**'. The IMAC L MePo List contains macros, which are required to conveniently edit MePo data.

The spreadsheet will open and automatically show the 'Edit' form as shown in Figure MePoList. The point selected for editing will be the point that has been active, when the Excel spreadsheet has been saved the last time (Active Row). If the active Excel row was not in the valid range of MePos currently contained in the list, the edit dialogue will jump to the first MePo in the List (if row-number was too small) or to the last entry in the list – an empty MePo (if row-number was too big).

For operations on 'spreadsheet level' the 'Edit' form has to be closed first.

**WARNING :** Worksheets, columns and headers in this spreadsheet may not be renamed or rearranged ! Otherwise the macros contained in the spreadsheet and the IMAC L data import to the AvET engineering tool will fail.



## 4.2 Worksheets of the IMAC L MePo Front-end

### 4.2.1 Worksheet 'Config'

This worksheet contains mainly IMAC L system settings, which **may not be modified in any way**.

Only three tables in this worksheet have to be filled in at the beginning of the MePo engineering:

- The table **'Section'** shall contain all Section ID's according to the system hierarchy. Numerical order (according to ID) is recommended. Any unique text identifier consisting of up to 6 characters is possible for identification, but the section naming convention as suggested above in this document is strongly recommended. The descriptive texts in the right column of this list are for easy user recognition of sections in the 'Edit' form only. The Section name "0" is reserved for system purposes (user definable section data list).
- The table **'TechID'** shall contain technology description texts for all 16 possible technology IDs of a project. The TechID numbering itself has to be in a range of '00' ... '16' and may not be altered.
- The table **'AIPI'** shall contain all graphics display identifiers in the IMAC L HMI, which are to be used as ALARM IDENTIFICATION PICTURES. The entry 'NOAIPI' may not be deleted. The descriptive texts in the right column of this list are for easy user recognition of sections in the 'Edit' form only.

In these tables no double entries and no empty entries shall be allowed.

If there are already MePos in the MePo list, and changes to these tables are required later, care has to be taken, that the assignment of TechID and Section number is still correct for all MePos. These tables just hold the list of possible entries for 'Section / Tech ID' in the 'Edit' form. If these have once been assigned to a MePo, the respective MePos will not change automatically, if this table will be modified later ! It is therefore strongly recommended to fill in these tables as the very first step of a project !

For the design of the system hierarchy please refer to [→2.1 Technology Areas, SECTION DATA and Process Graphics Displays, page 9].

13		17		1	
Section		TechID		AIPI	
00	Free definable Section	00	Void	NOAIPI	No AiPi assigned
0101	One Line Diagram	01	Electrical Systems		
0102	Power Management System	02	Main Engine 1		
0201	Main Engine 1 (general)	03	Main Engine 2		
0202	ME 1 Exhaust Gas	04	Main Engine 3		
0601	Purifiers	05	Main Engine 4		
1401	ER Bilge System	06	Auxilliary Engines		
1402	Cargo Bilge Sys	07	Sea Cooling Water System		
1501	ER Ventilation	08	Fresh Cooling Water System		
1502	Car Deck Ventilation	09	Aux Diesel / Generator		
1503	Galley Ventilation	10	Fuel System		
1504	Accommodation Ventil	11	Boiler & Steam		
1601	IMAC System	12	Cargo Tank/Hold		
		13	Ships Safety Systems		
		14	Bilge & Ballast Systems		
		15	HVAC		
		16	IMAC		

**Figure MePoSheetConfig** shows the three IMAC L Config tables, that have to be modified by the user

#### 4.2.2 Worksheet 'Objekte'

The worksheet 'Objekte' contains the basic MePo list of IMAC L. This spreadsheet will contain one row for each MePo starting with row 6 for the first MePo (rows 1-5 are header – do not modify).

Every MePo requires a unique Identifier (PointID, i.e. name) in IMAC L.

In this worksheet all MePos in the list have to be in one block starting from row 6 in the spreadsheet. Empty rows are not permitted. The MePo list will distinguish 'empty' rows from rows filled with MePo data by the contents of the 'PointID' field. Dummy PointIDs shall be allowed (the 'Edit' form itself will use the dummy 'new' for every inserted MePo. Nevertheless the 'Check' functions will identify duplicate dummy PointIDs as failures (which is a good reminder to either remove this dummy or to fill it in correctly).

If MePo editing is done via the 'Edit' form only, empty rows will be avoided automatically.

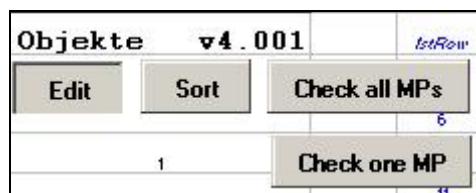
When the 'Edit' form is started, it will check for 'empty' rows (containing no PointID) within the block of MePos in the list. If empty rows are detected, the form will suggest to sort the MePo list by the 'PointID' column to remedy this. If the user does not accept this suggestion, the user will have to manually remove all empty lines or enter valid PointIDs to empty rows before the 'Edit' form can be used.

All possible parameters for every MePo will be stored in 142 columns. The basic naming of the columns, the respective naming in the 'Edit' form and a text describing each type of column can be found in the table below :

Column Name	Name in Edit form	Description
AAGRx	AAG	Audible Alarm Group of status 'x'
AdrValue	Address	Address for MP data in S7-format
AIPI	AiPi	Alarm Identification Picture that has been assigned to this MP
AIVI	AiVi	Alarm Indicating Video that has been assigned to this MP
ALx	A Lvl	Alarm level of status 'x'
BLGx	BLG	Blocking Group of status 'x'
CompID	CompID	ID of Compensating Measuring Point
DelInx	Delay In	Time Delay 'In' for entering status 'x'
DelOutx	Delay Out	Time Delay 'Out' for leaving status 'x'
Description	Description	Measuring Point Description
DyLiGrpx	Dyn Limit Group	Dynamic Limit Group for Intervals 1 and 2
EVL	EVL	Checkmark 'Event Logging On' for this MePo
EVPx	EVP	Checkmark 'Event Printing On' for status 'x'
FNoAckn	FNoAckn	Function Number for Acknowledge (user access rights)
FNoChange	FNoChange	Function Number for Changes (user access rights)
Format	Format	Numerical format of a PV (used for 'PV' only)
HITI	HiTi	Hints and Tips text that has been assigned to this MP
InitValue	InitValue	Checkmark 'Use default value for initialisation'
LastChange	Last Change	Date & time of last Change
LimitHyst	Limit Hyst	Limit Hysteresis (in %) for Detection of Limit Infringements
Limitx	Limit(s)	Limit Value between Intervals x and x+1
MPTYPE	MP-Type	Type of measuring Point (analog, binary)
NoOfStatus	No Of Status	Number of status conditions the MP can have
ObjClass	Obj-Class	Class of this object (MP or PV)
PointID	Point ID	Measuring Point Identifier, unique name of the MePo
PVType	PV-Type	Type of Process variable (numerical, string)
RangeEnd	Range End	End of Range
RangeStart	Range Start	Start of Range
Remark	Remarks	Remarks
ScalGrp	ScalGroup	Scaling Group number
Section	Section Data	Section number to which this MP is assigned
SendHyst	Send Hyst	Sending Hysteresis (in %) used for detection of changes
SOGRx	SOG	Signal Output Group of status 'x'
Station	Station	Process Control Station (PCU) to which this MP is assigned
StsTextx	Status Text	Status Text of status 'x'
TechID	TechID	Technology Area ID of
Terminal	Terminal	Description of Connection Point (Slot & Terminal)
Transmitter	Transmitter	Transmitter Type for input / output
UndefGrp	UndefGrp	Undefined Group that has been assigned to this MP
Unit	Unit	Physical Unit used
User	User	Name of the user who performed last change to this MP
VAGRx	VAG	Visual Alarm Group of status 'x'
ValueDefault	Value (default)	Start Value for new start / re-start
Xtype	X-Type	Type of X-Variable (numerical format used)



In the upper left corner of the worksheet 'Objekte' four buttons can be found :



The functions of these buttons are as follows :

Item	Description
Button 'Edit'	<p>Open the 'Edit form'. The first point selected for editing will be the point that is active when pressing 'Edit' (Active Row). If the active Excel row was not in the valid range of MePos currently contained in the list, the edit dialogue will jump to the first MePo in the List (if row-number was too small) or to the last entry in the list – an empty MePo (if row-number was too big).</p> <p>The edit form, its use and input fields will be discussed in detail later on in this document.</p>
Button 'Sort'	<p>The MePo list can be sorted by any of the columns by selecting one of the cells in this column and then pressing the 'Sort' button. A dialogue will ask the user, if this operation is really desired. Pressing OK in this dialogue will initiate the sorting.</p> <p>If the naming scheme and rules as described above in this document are followed, the alphabetical sorting to the 'PointID' will result in a neat assortment of MePos.</p> <p><b>Attention :</b> If you have created a 'manual' order within your MePo list, sorting will irreversibly destroy this manually created sequence by assorting MePos alphabetically according to the selected column.</p>
Button 'Check all MP's'	<p>Check all MePos data sets in the list for basic consistency. A window will open displaying the progress of this check and its results. For MePo lists with several hundreds of entries, the checking may take some time. The results are displayed as text lines (one line per faulty MePo). The Fault message will refer to the row number in the Excel sheet and the MePos PointID. The field(s) considered as faulty will be listed as well.</p> <p>The check function will automatically highlight the faulty MePos in the spreadsheet itself (yellow background, red font). The first row ('PointID') of a faulty MP will always be highlighted (to easily find faults in the list). Additionally all fields in this row which have been found to be faulty will be highlighted as well.</p> <p>The results can be printed (screen shot, print to Windows default printer). For an example see Figure MePoCheckAll below.</p>
Button 'Check one MP'	<p>Will just check one MePo (the one contained in the row currently selected). The check is identical to 'Check all MP's', but will not display a result window. The result indication is just via the colour changes of the current row (same as 'Check all MP's').</p> <p>This function is useful in large MePo lists, where direct editing in the spreadsheet is performed. After modifying a MePo the changes can be directly verified using this function.</p>

The 'Check all MP's' function will open a report window as shown below:

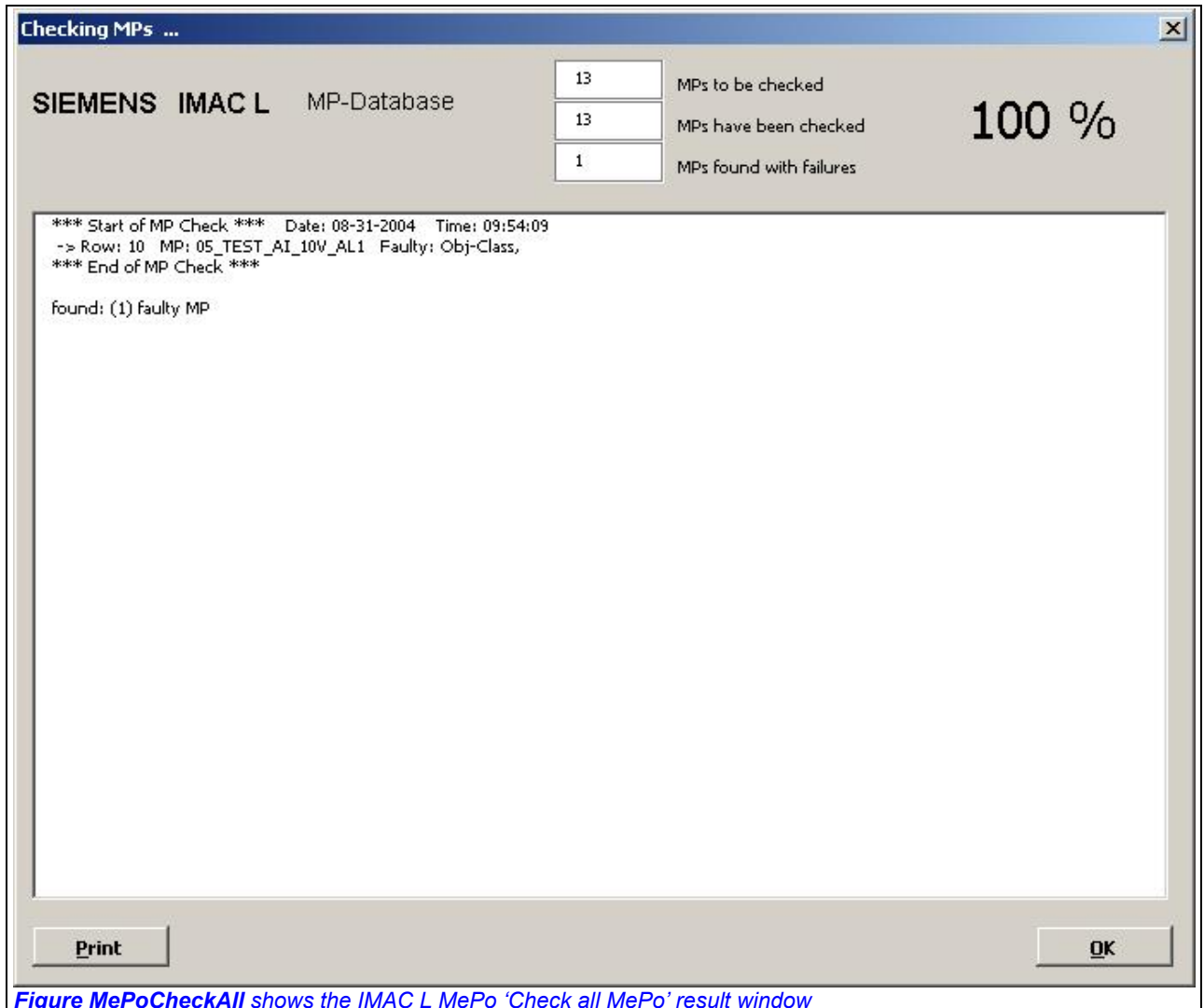


Figure MePoCheckAll shows the IMAC L MePo 'Check all MePo' result window

The 'Check one MP' function will show faults in the respective MePo row with colour marks :

3	1	Check one MP		4	5
4			II	0	0
5	Identifier	ObjClas:	MPTYPE	PVType	Xtype
6	01_TEST_DI	MP	binär	Num	ANY
7	02_TEST_DI_AL	MP	binär	Num123	ANY
8	03_TEST_DI_AL_BL	MP	binär	Num	ANY

Figure MePoCheckOne shows the IMAC L MePo 'Check one MP' result as colour marks (here: PV Type is unknown)

## 4.2.3 Worksheet 'Status Texts'

This worksheet is used to collect Template 'Status Texts' used in IMAC L. This list can be manually filled in, if an appropriate list of status texts has been agreed upon for a project.

Status Texts			Zeile1	AnzDaten
Sort	Check	Delete Duplicates	5	9
Status Text	Source	User	Last Change	
High	SIEMENS	57125400@DE2ZEV6D	11.08.2004 12:16	
Low	SIEMENS	57126994@DE2ZERCD	16.07.2004 09:30	
Normal	SIEMENS	57126994@DE2ZERCD	16.07.2004 09:30	
Normal2	SIEMENS	57125400@DE2ZEV6D		
OFF	SIEMENS	57125400@DE2ZEV6D	12.08.2004 14:23	
ON	SIEMENS	57125400@DE2ZEV6D	12.08.2004 14:23	
Out Of Range	SIEMENS	57126994@DE2ZERCD	16.07.2004 14:57	
Very High	SIEMENS	57125400@DE2ZEV6D	12.08.2004 14:39	
Very Low	SIEMENS	57125400@DE2ZEV6D	12.08.2004 14:41	

Figure MePoSheetStsTxt shows worksheet 'Status Texts'

All entries in this list will be offered as possible 'predefined' selections in the appropriate 'combo boxes' in the 'Edit' form. This list is meant to ease the assignment of status texts by avoiding to re-type the same texts again and again.

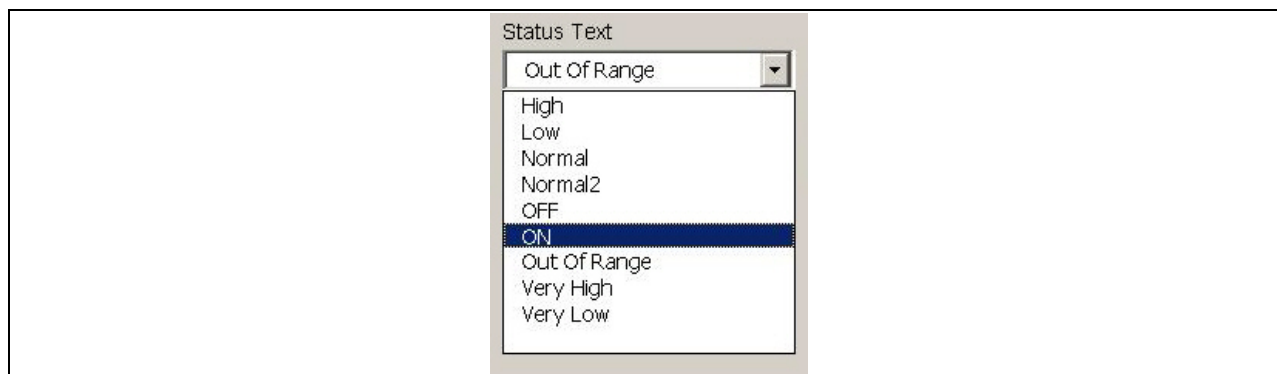


Figure MePoSheetStsCombo Shows a 'Combo Box' in the 'Edit' form showing the listed status Texts

If double or undesired entries have been created, entries may be modified or deleted as desired. Modifying this list will not change any existing MePo list entries. Every MePo holds its own copy of these texts. This list is just to simplify entering new status texts.

If in the 'Edit' form a new status text is entered for any MePo status, the user will be asked, if this new text is to be added to the 'Status Texts' list. Answer 'Yes', if you expect this status text to be required multiple times. Answer 'NO' for unique or exotic texts not expected to be re-usable.

The columns 'Source', 'User' and 'Last change' are automatically filled in, if new texts are entered via the 'Edit' form. This is meant to keep track of changes to this list. The contents of these columns are not required for the function of the 'Edit' form. Entries added or modified manually do not necessarily need these columns to be filled in.

If new texts have been added, sorting using the **'Sort' button** is recommended. This will present the status texts in the 'Edit' forms combo boxes in an alphabetical order.

Pressing the **'Check' button** will highlight invalid (duplicate) entries.

Pressing the **'Delete Duplicates' button** will sort this list alphabetically and then remove duplicate entries automatically.

## 4.2.4 Worksheet 'Status'

This worksheet is for information only (no function). It has been included to help the user understand the Alarm Levels and Status used in IMAC L.

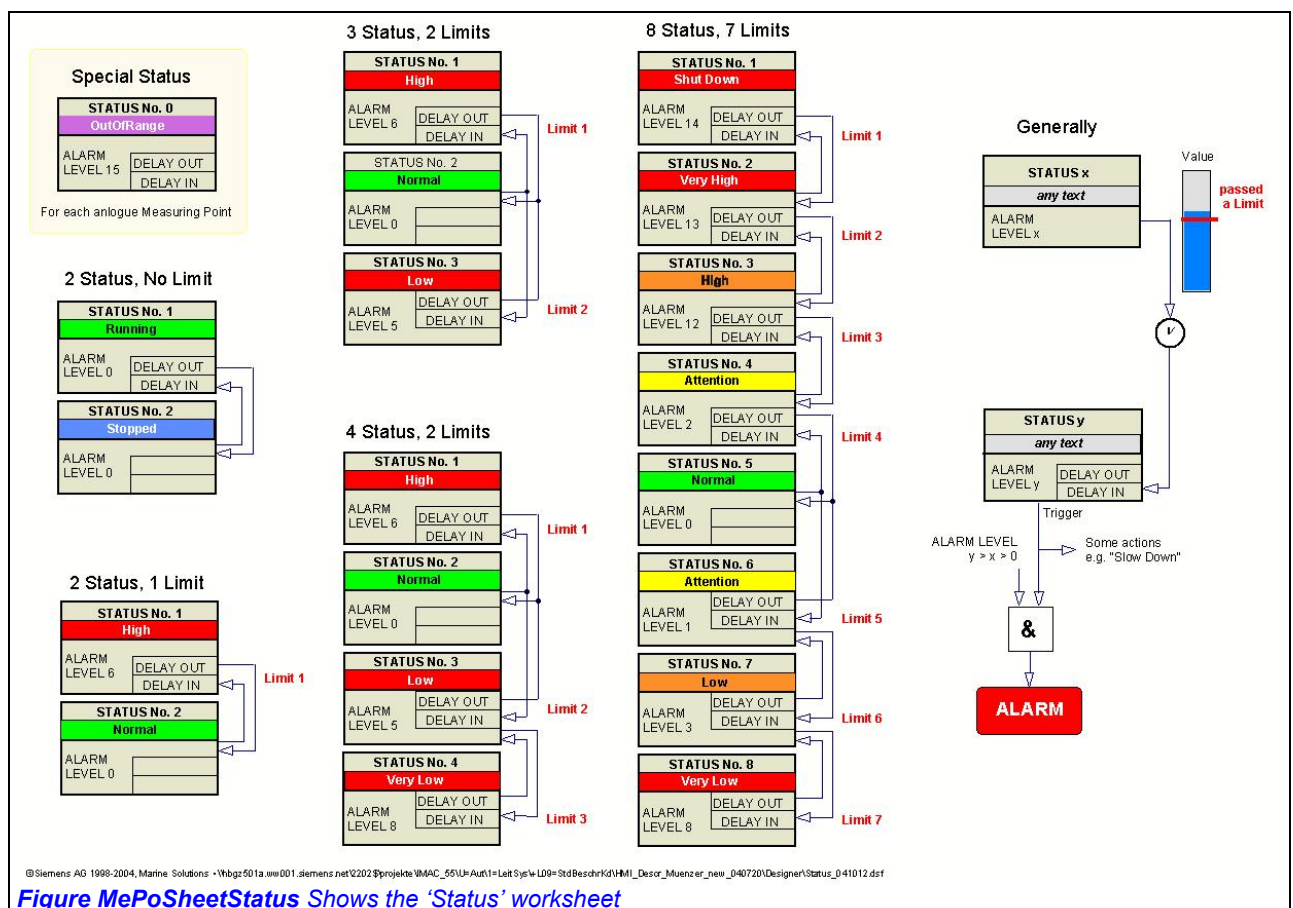


Figure MePoSheetStatus Shows the 'Status' worksheet

### 5. Creating and Modifying MePos with the 'Edit' Form

The 'Edit' form has the following basic appearance :

The screenshot shows the 'Objekt' window for editing a MePo. The form is divided into several sections:

- Point ID:** 010105
- Obj-Class:** MP
- MP-Type:** analog
- PV-Type:** (empty)
- Description of Point (max 30 Characters):** Sample Point Analog (4 Limits)
- Station:** PCU1
- Section Data:** 0101
- TechID:** 01
- AIPI:** NoAIPI
- AIVi:** NoAIVi
- HiTi:** NoHiTi
- UnderGrp:** 2
- FNoAckn:** (empty)
- FNoChange:** (empty)
- EVL:** ☒
- EVP:** ☒
- Address:** PEW 272
- Value (default):** ?
- InitValue:** (empty)
- Transmitter:** AI +-0...10V
- Terminal:** (empty)
- Format:** REAL
- Range Start:** 0
- Range End:** 500
- Unit:** DegC
- Send Hyst:** 0.5
- Limit Hyst:** 2
- Compld:** (empty)
- ScalGroup:** (empty)
- No Of Status:** 5
- Table of Limits:**

#	A Lvl	BLG	VAG	AAG	SOG	Status Text	Delay In	Delay Out	Limit(s) in DegC	Dyn Limit Group
0	15		3	1		Out Of Range	1	1		
1	13		3	1		Very High	2	2	470	
2	12		3	1		High	1	1	460	
3	0		0	0		Normal	1	1	420	
4	2		3	1		Too Low	1	1	410	
5	8		3	1		Very Low	1	1		
6										
7										
8										
- Remarks:** (empty text area)
- Last Change:** 23.06.2009 12:16:58
- Excel Row No:** 20
- User:** 57125400@E51A2KSD
- Buttons:** Previous, Move UP, Insert bef., Discard Chg., Save, Print, Goto Row, Next, Move DOWN, Insert after, Delete, Save as, Close.

Figure MePoEditAnalog shows the IMAC L MePo 'Edit' Form (in this example an analogue MePo / 6 AI-Levels)

Depending on the type of MePo and its properties only those fields will be displayed, which are relevant for this type and setting of MePo.

The basic fields influencing this 'show / hide' function of fields in the form are :

- Obj-Class
- MP-Type
- No Of Status



## 5.1 Basic Editing Functions


The Edit form has a set of basic indication fields and editing buttons in the lower part of the form :

Item	Description
Field 'Last Change'	Date and Time of last change to this MePo
Field 'User'	Username who performed the last change (Windows user name)
Excel Row No	Number of the row in this spreadsheet, which contains the data of this MePo.
Button 'Goto Row'	Requests to enter a row number. Navigates to the MePo contained in this row. If the row number entered does not contain a MePo, a fault message will be displayed. If the MePo data in the current form have been edited but not saved, the user will be asked, if changes are meant to be discarded. Save the current MePo before navigating to avoid this message.
Buttons 'Previous / Next'	Navigate to the previous / next MePo in the list. If the MePo data in the current form have been edited but not saved, the user will be asked, if changes are to be saved or discarded. Save the current MePo before navigating to avoid this message.
Buttons 'Move Up' / Move Down'	Manually Sort a MePo list by moving the current MePo one row up / down in the list.
Buttons 'Insert bef.' / 'Insert after'	Insert a new (empty) MePo before or after the current MePo and navigate to this MePo. The MePo PointID will be set to the default name 'new'. The new MePo will be indicated as 'faulty' by default, because it is missing basic settings. If the MePo data in the current form have been edited but not saved, the user will be asked, if changes are meant to be discarded. Save the current MePo before inserting to avoid this message.
Button 'Discard Chng.'	Discard changes made to the MePo in this form. The user will be prompted, if this operation is really desired.  Basic function of 'Discard Chng.': if a MePo is loaded, the 'Edit' form will copy all values from the spreadsheet cells of the MePo to the forms fields. Pressing 'Save' or 'Save as' will store these values back to the spreadsheet cells. Pressing 'Discard' will ignore all values which have been entered to the form and will instead re-load the values of this MePo as it is currently stored in the spreadsheet cells.
Button 'Delete'	Completely delete a MePo (results in irreversibly deleting one row in the list, losing all information in this row / MePo). The user will be prompted, if this operation is really desired.

Item	Description
Button 'Save'	<p>Saves the values in the form back to the spreadsheet cells of the respective MePo. 'Save' is only allowed, if the MePo name is valid (Field 'PointID' is not empty, does not consist of 'spaces only' and there is no other MePo in the MePo list having the same name).</p> <p>The MePo name has to be unique within a IMAC L system.</p> <p>Please observe : 'Save' will store the values in the 'Edit' form back to the Excel cells belonging to the respective MePo. <b>This operation does NOT save the Excel Spreadsheet ! It is strongly recommended to save the Excel Spreadsheet in regular intervals during your work !</b> To save the Excel spreadsheet close the 'Edit' form and save the spreadsheet as desired. Then restart the 'Edit' form by pressing the 'Edit' button in the sheet again.</p>
Button 'Save as'	<p>Like 'Save', but used to append a new MePo to the end of the MePo list. The 'PointID' has to be unique in IMAC L.</p> <p><b>Tip :</b> this function can be used to duplicate a MePo by loading an existing MePo into the form. Then modify the MePos PointID to a new, unique name and press 'Save as'.</p>
Button 'Print'	Will print a copy of the current MePo. The printer used is the windows default printer with its default settings.
Button 'Close'	Close the 'Edit' form. If the MePo data in the current form have been edited but not saved, the user will be asked, if changes are meant to be discarded. Save the current MePo before closing the edit form to avoid this message.

## 5.2 Fault Indications in the Edit Form

The 'Edit form will check values entered to its fields for basic consistency. Every field for which a fault has been detected will immediately be highlighted (red font on a yellow background) :




The screenshot shows a form with four input fields: 'Address' (containing 'DB100.DBW2'), 'Value (default)' (empty), 'Range Start' (containing '0'), and 'Range End' (containing '100a'). The 'Range End' field is highlighted with a yellow background and red text, indicating a fault. Below the form, a caption reads: *Figure MePoCheckEdit Shows a field with a highlighted fault (letter 'a' has been entered in a 'number' type field)*

MePos can be saved, even if they contain faults (exception: missing or double 'PointID'). Nevertheless all failures have to be taken care of before starting to generate a IMAC L system from these data.

Failures that have been detected by the 'Edit' form will be highlighted in the 'Objekte' worksheet in the same instant (to make them visible for editing on the spreadsheet level).

### 5.3 Description of all Input Fields in the Edit Form

#### 5.3.1 Field : 'PointID'

	
Full name	Point Identifier
Type of input	Text input (ASCII Characters). Text Length : max. 8 Characters
Spreadsheet Column	A

The PointID will unambiguously identify a measuring point in IMAC L. Within one IMAC L system every PointID has to be unique. A PointID may consist of numeric or alphabetic characters or of a combination of both. Generally all ASCII characters are permitted, but country specific characters and special characters shall be avoided.


For any HMI user input of PointIDs IMAC L will not distinguish between 'upper case' and 'lower case' characters. Nevertheless it is good practice to have a consistent use of upper / lower case for PointIDs (and all other texts and IDs used in the system) in order to achieve a uniform 'touch and feel'.

MePo PointIDs beginning with an underscore character (" \_ ") are reserved for IMAC L system internal points.

[More information on PointIDs →

2.1.2 Structuring, Identification for Measuring Points, page 11].

#### 5.3.2 Field : 'Obj-Class'

	
Full name	Object Class
Type of input	Selection using a 'combo box'. Allowed choices : "MP", "PV"
Spreadsheet Column	B

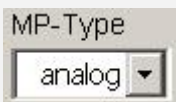
In IMAC L generally two types of points have to be distinguished :

- MP = Measuring Points. Used to collect ('measure') information in the PCUs. This information is later on used to display values and resulting alarms on Operator Stations (OSs). This is the most common type used.
- PV = Process Variables. Process variables are used to transport user control input from OSs to PCUs (e.g. setpoints, control commands)

Every point has to be assigned to one of these two types. The selection of the object class will show / hide several fields in the 'Edit' form, because a PV will require only a small sub-set of input compared to a MP. Therefore during input of new measuring points this field should be filled in as one of the first fields.



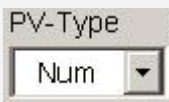
**5.3.3 Field : 'MP-Type'**

	
Full name	Measuring Point Type
Type of input	Selection using a 'combo box'. Allowed choices : "analog", "binär"
Spreadsheet Column	C

This field will be visible for points with object class 'MP' only.

Each 'MP' measuring point has to be either analog or binary ("binär"). The selection of the object class will show / hide several fields in the 'Edit' form, because a binary point will require only a subset of the input required for analog points. Therefore during input of new measuring points this field should be filled in as one of the first fields.

**5.3.4 Field : 'PV-Type'**

	
Full name	Process Variable Type
Type of input	Selection using a 'combo box'. Allowed choices : "Num", "Str"
Spreadsheet Column	D


This field will be visible for points with object class 'PV' only.

In IMAC L generally two types of PVs have to be distinguished :

- Num = Numerical Value
- Str = String Value

Most control information will be numerical values (type 'Num'). The string type ('Str') will only be required, if user text input has to be transferred to a PCU.

**5.3.5 Field : 'Description'**

	
Full name	Description of Point
Type of input	Text input (ASCII Characters). Text Length : max. 30 Characters
Spreadsheet Column	F

This text serves as a short description of the measuring point. Basically any ASCII character can be used. This text (and the PontID) will be displayed in the HMIs ALARM SUMMARY, POINT REPORT, EVENT LOG and other MePo related displays. Due to the limited length of this description text, abbreviations will usually be necessary.

For more information on the design of MePo description texts see [→3.1 Measuring Point Description Texts, page 16].

For more information on the use of abbreviations see [→ 3.1 Measuring Point Description Texts].

### 5.3.6 Field : 'Station'

<div>Station</div> <div>PCU1 ▾</div>	
Full name	Process Control Station
Type of input	Selection using a 'combo box'. Allowed choices : "PCUx", x=1....9
Spreadsheet Column	G

This field will assign a MePo to one of the Process Control Stations (PCU) in the IMAC L system. As a rule this is the PCU which holds the hardware (I/O-module) to acquire this signal. For the PV type of point this is the PCU who is to receive the control input from this PV.

### 5.3.7 Field : 'Section Data'


<div>Section Data</div> <div>1601 ▾</div>	
Full name	Section Data Identifier
Type of input	Selection using a 'combo box'. Choices according to list in sheet 'config'
Spreadsheet Column	H

Name of the SECTION DATA list to which a MePo belongs. A SECTION DATA list is a compilation of MePos, which have a logical and / or technological relation to each other. All MePos in one SECTION DATA list can in the HMI be viewed as a separate list called a SECTION DATA list or can be printed as a journal.

The section naming is part of the system hierarchy design. For more information on design and naming of sections in an IMAC L system see [→2.1 Technology Areas, SECTION DATA and Process Graphics Displays, page 9].

All names of all sections have to be filled in to the list in sheet 'Config' prior to begin of editing MePos. For more information on the section list in the sheet 'Config' see [→4.2.1 Worksheet 'Config', page 26]. All entries in this list will be available as a possible choice in the 'Section' combo box.

**5.3.8 Field : 'TechID'**


	
Full name	Technology Area Identifier
Type of input	Selection using a 'combo box'. Choices according to list in sheet 'config'
Spreadsheet Column	EC

Name of the TECHNOLOGY AREA to which a MePo belongs. A TECHNOLOGY AREA is the major grouping level for the hierarchical structure of IMAC L. MePos functionally belonging together will be grouped in the TECHNOLOGY AREAS. Every TECHNOLOGY AREA has one button assigned to it in the IMAC L HMI header area giving access to the corresponding graphics displays and indicating group alarms with colour changes of the buttons.

The TECHNOLOGY AREA naming is part of the system hierarchy design. For more information on design and naming in the hierarchy of IMAC L see [→2.1 Technology Areas, SECTION DATA and Process Graphics Displays, page 9].

All IMAC L systems will have either 16 or 8 technology areas. All description texts for these 16 (8) TECHNOLOGY AREAS have to be filled in to the list in sheet 'Config' prior to begin of editing MePos. The TechID numbers have to be consecutive and may not be modified in the sheet 'Config'. For more information on the section list in the sheet 'Config' see [→4.2.1 Worksheet 'Config', page 26]. All entries in this list will be available as a possible choice in the 'TechID' combo box.

**5.3.9 Field : 'AiPi'**


	
Full name	Alarm Identification Picture
Type of input	Text input (ASCII Characters). Text Length : max. 6 Characters
Spreadsheet Column	EF

The ALARM IDENTIFICATION PICTURE 'AiPi' will be used to easily navigate from a list (e.g. ALARM SUMMARY, POINT REPORT) to an existing process graphics display. IMAC L will interpret this text as a graphics display identifier ('GD-ID'). A graphics display with the name as entered in this field has to exist in the IMAC L HMI. Clicking in the AiPi button will immediately branch to the graphics display with the name as entered in this field.

Entering 'NOAIPI' will disable the AiPi function for this MePo.

All AiPis to be used have to be filled in to the list in sheet 'Config'. For more information on the AiPi list in the sheet 'Config' see [→ 4.2.1 Worksheet 'Config', page 26]. All entries in this list will be available as a possible choice in the 'AiPi' combo box.

## 5.3.10 Field : 'AiVi'

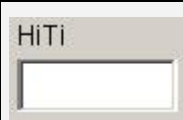
	
Full name	Alarm Identification Video
Type of input	Text input (ASCII Characters). Text Length : max. 6 Characters
Spreadsheet Column	EH

The Alarm Identification Video 'AiVi' will only be required for IMAC L systems equipped with a video monitoring system. MePos that have been assigned to a video source will have an AiVi button in their POINT REPORT and in their ALARM SUMMARY entry. The AiVi field will specify for a MePo, which video source will be selected, if the AiVi button of this MePo is pressed.

Entering 'NOAiVi' will disable the AiVi function for this MePo.

The AiVi names entered in this field have to be consistent with the engineering of the video monitoring system (i.e. the video monitoring system will use this text string to identify the correct video source).

## 5.3.11 Field : 'HiTi'

	
Full name	Hints & Tips Text
Type of input	Text input (ASCII Characters). Text Length : max. 6 Characters
Spreadsheet Column	EG


Reference to a 'Hints & Tips' text assigned to this MePo. A HiTi is a HTML file, which will be displayed in an AbHelp window, if the user clicks on the HiTi button (e.g. in the POINT REPORT) or on 'F1' (optional). HiTi will only work for those MePos with a valid entry in this field. One HiTi text can be referenced by one or more MePos. Only measuring points (MPs) can have a HiTi text assigned to them, but not process variables (PVs).

An identifier for a HiTi file can consist of up to 6 ASCII characters. Only characters and digits but no special characters are allowed in this field. The HiTi-ID entered in this field is not directly the name of the HTML file to be displayed. For more information on 'HiTi' engineering see → 11.4 IMAC L 'Hints & Tips' (HiTi), page 196.

Like for Point IDs a systematic naming of HiTi-Ids and associated HTML files throughout an IMAC L system is recommended.

If this field is left empty, no HiTi text will be available for the corresponding MePo.


## 5.3.12 Field : 'UndefGrp'

	
Full name	Undefined Group
Type of input	Numerical Value. Range : 1 ... 32 (leave field empty, if not applicable)
Spreadsheet Column	U

UNDEFINED GROUP [→ 9.1 Undefined Groups on page 106] for signals received e.g. from serial lines or bus communication links. Measuring Points derived e.g. from a specific serial interface line will have to be assigned to the same UNDEFINED GROUP 'x'. If IMAC L will detect, that this serial line has a communication fault, all MePos from this interface (i.e. all MePos with this UNDEFINED GROUP 'x' in its properties) will be automatically set to the status Undefined.

If this field is left empty, the MePo will not be member of an undefined group. Usually this should only happen, if an I/O module is directly connected to a CPU. IMAC L will use 'ET 200 S' Profibus slaves for the majority of it's I/Os. Every Profibus Slave will require its own Undefined Group for proper indication of data in case of Profibus communication faults.

## 5.3.13 Field : 'FNoAckn'

	
Full name	Function Number for Acknowledge
Type of input	Numerical Value. Range : 0 ... 99
Spreadsheet Column	EJ


Every measuring Point with an alarm level has to be assigned to a function number for acknowledge. This function number will later on determine, which users are allowed to acknowledge alarms resulting from this MePo. Only those users having this function number either in their user profile or in their active SIC profile will be allowed to acknowledge alarms resulting from this MePo.

As a simple case a ship can have only one function number for acknowledging alarms. Every SIC / user profile of users requiring rights to acknowledge alarms will have this function number assigned to it.

If a zero ('0') is entered to this field, acknowledging of alarms from this MePo is allowed for any user without restrictions.

For more detailed information on the concept of user profiles and the SIC (Station In Control) concept please refer to the document 'IMAC 55 Human Machine Interface (HMI) Description' (document number E10231-Y1009-U010).

## 5.3.14 Field : 'FNoChange'

	
Full name	Function Number for Changes
Type of input	Numerical Value. Range : 0 ... 99
Spreadsheet Column	EK


For every measuring Point with parameters (limits, delays) a function number for acknowledge has to be set. This function number will later on determine, which users are allowed to change the parameters of this MePo. Only those users having this function number either in their user profile or in their active SIC profile will be allowed to change parameters to this MePo.

As the simplest case a ship can have only one function number for changing parameters. Every SIC / user profile of users requiring rights to acknowledge alarms will have this function number assigned to it. It is common practice to put this function number to the user profile of the chief engineer only.

If a zero ('0') is entered to this field, changing parameters of this MePo is allowed for any user without restrictions.

For more detailed information on the concept of user profiles and the SIC (Station In Control) concept please refer to the document 'IMAC 55 Human Machine Interface (HMI) Description' (document number E10231-Y1009-U010).


## 5.3.15 Field : 'EVL'

	
Full name	Event Log
Type of input	Checkmark ('WAHR' (true) / 'FALSCH' (false) in spreadsheet)
Spreadsheet Column	AI

This Checkmark determines, if status changes for this MePo will be entered to the IMAC L Event-Log. Checking this box is a prerequisite for all MePos that are meant to create an alarm of any kind (this usually includes all analogue MePos having an 'Out of Range' status). Only those MePos just designed for indication (e.g. ON/OFF) will not require this checkmark to be set.

A MePo having this checkbox 'not set' will not show up in the IMAC L ALARM SUMMARY and EVENT LOG lists and will not be able to cause an alarm.

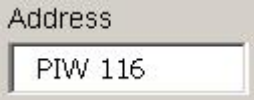
### 5.3.16 Field : 'EVP'

	
Full name	Event Log Print
Type of input	Checkmark ('WAHR' (true) / 'FALSCH' (false) in spreadsheet)
Spreadsheet Column	AJ

This Checkmark determines, if a status change will be printed on the Event Log Printer.

Printing events (EVP = Wahr/True) requires the EVL-checkmark of this MePo to be set as well.


### 5.3.17 Field : 'Address'

	
Full name	Simatic S7 Address
Type of input	Text describing a valid Simatic S7 data element of the appropriate type
Spreadsheet Column	J

Text describing a valid data source in the Simatic S7 PLC. This can be a direct I/O address or an address referring to a data element in a data block. The notification and language of this address has to be in GERMAN :

Data Type	GERMAN Step 7 (example)	English notation
Digital Input Bit	E x.y	I x.y
Digital Output Bit	A x.y	Q x.y
Digital Input Byte	EB x	IB x
Digital Output Byte	AB x	QB x
Digital Input Word	EW x	IW x
Digital Output Word	AW x	QW x
Data Block Bit	DBx.DBXy.z	DBx.DBXy.z
Data Block Byte	DBx.DBBy	DBx.DBBy
Data Block Word	DBx.DBWy	DBx.DBWy
Data Block Double Word	DBx.DBDy	DBx.DBDy

### 5.3.18 'Smart Help' Button : '?'

	
Full name	? (called 'Smart Help')
Type of input	Button-Click
Spreadsheet Column	none (modifies input in column J)

This 'Smart Help' button is an aid, which was designed to help filling in pre-defined data addresses for IMAC L process objects (e.g. Motors, Valves), which are required to control & monitor such process objects.

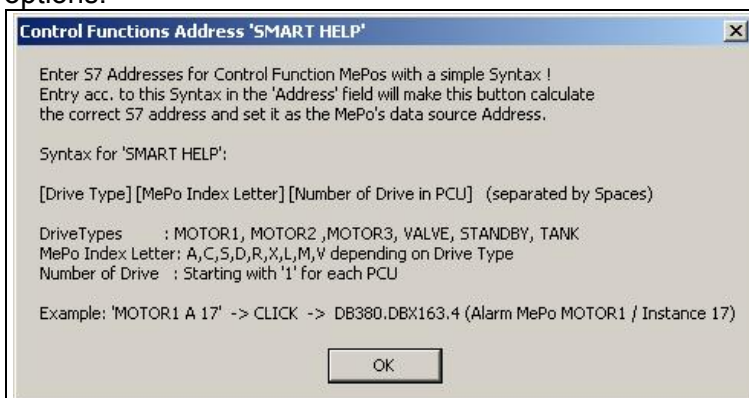


All IMAC L absolute data addresses to monitor & control such objects are predefined. See section → '11.7.1. MePos required for HMI access to 'Control Function' Objects' for more information on this type of data objects and their allocation. The operator has to precisely select the right data object address in a Data Block in order to make such a objects work. This can be a difficult task which is prone to error, as the entered information in field '5.3.17 Field : 'Address' is an absolute Simatic data address (and not something symbolic).

The 'Smart Help' ('?') button makes it possible to enter a systematic reference to such control object in the field 'Address'. The entry has to be according to a predefined syntax.

If the syntax entered in the field 'Address' was recognized, the function behind this button will automatically resolve the required address and replace the entered syntactical string in the field 'Address' with the correct Simatic address for the desired object.

If the entered syntax was not recognized correctly, a popup window will display the possible syntax options:



**Figure SmartHelp Shows the PopUp Window showing the correct syntax for 'Smart Help'**

### Example:

- The MePos for IMAC L process object '**010203**' of type '**MOTOR1**' are to be defined.
- The object shall be the IMAC L '**MOTOR1**' instance # '**17**' (the 17<sup>th</sup> call in this PLC)

To set the correct address for the IMAC L 'Alarm Status Word' (MePo ID '**010203**')

Enter '**MOTOR1 17**' in the 'Address' field →

Address

MOTOR1 17 ?

Click on 'smart Help' (?) →

Address

DB380.DBW166 ?

→ DONE !

In the same style, the entry for all four required '**MOTOR1**' data objects for '**010203**' would be:

MePo ID	Description	→	Entry Syntax	→	Resolved Address
<b>010203</b>	Alarm Status Word	→	' <b>MOTOR1 17</b> '	→	DB380.DBW166
<b>010203A</b>	Alarm Flag	→	' <b>MOTOR1 A 17</b> '	→	DB380.DBX163.4
<b>010203C</b>	Control Word	→	' <b>MOTOR1 C 17</b> '	→	DB380.DBW168
<b>010203S</b>	Status Word	→	' <b>MOTOR1 S 17</b> '	→	DB380.DBW164

### 5.3.19 Field : 'Value (default)' & 'InitValue'

<div>Value (default)    InitValue</div> <div> <input type="text" value="0"/> <input checked="" type="checkbox"/> </div>	
Full name	Value (default) & Initialize Value
Type of input	Numerical Value (Value default) and Boolean (InitValue)
Spreadsheet Column	M, L

Initializing value to be applied to the MePos data source during startup of the PCU. This function may only be used for internal MePos (i.e. those MePos who do not have a data source resulting from hardware I/O). This is especially useful for MePos containing data from an internal computation. As long as this computation has not yet been performed, the initial value will be presented in IMAC L (because this value has been copied to the data source address during startup of the PLC). As soon as the computation has been performed once, the initial value will be replaced with the real value.

To use this function, enter a numerical value as desired to the field 'Value (default)' and place a tick to the 'Init Value' field. The value field may be left empty, if the 'InitValue' field is not checked.

### 5.3.20 Field : 'Transmitter'

<div>Transmitter</div> <div> <input type="text" value="AI +-0..10V"/> </div>	
Full name	Transmitter Type
Type of input	Selection using a 'combo box'. Choice of all predefined types in IMAC-L
Spreadsheet Column	I

Type of transmitter used for this MePo. This setting has to match the 'Address' field setting used for this MePo. If a MePo refers to an 'Address' in the Simatic S7, the Simatic hardware configuration or software source (data block) has to be set in such a way, that a value of type 'Transmitter' can be processed with this source 'Address'. For internal MePos (e.g. from a S7-data block) special transmitter types are available.

### 5.3.21 Field : 'Terminal'

<div>Terminal</div> <div> <input type="text" value="X210.23-24"/> </div>	
Full name	Terminal / Slot (point of electrical connection)
Type of input	Text input (ASCII Characters). Text Length : max. 20 Characters
Spreadsheet Column	EI

Text describing the point of the electrical connection of this MePo (Rack / Slot / Terminal). Only necessary for MePos who have a data source resulting from hardware I/O. The naming of this terminal information shall be consistent with the electrical plans delivered to the customer.

This text will be displayed in the Point Report of the IMAC L HMI (field 'PATH') and will offer to the operator an easy-to-understand reference to the hardware related to this MePo. This is especially useful for trouble shooting in IMAC L systems (sensor failures are the most common failure on board of a ship).

### 5.3.22 Field : 'Format'

<div>Format</div> <div>REAL</div>	
Full name	Format
Type of input	Selection using a 'combo box'. Choice of predefined types for MP or PV
Spreadsheet Column	N

Input of the Value format to be used. Has to be entered for all points of type 'MP analog' and type 'PV Num'. Available choices in the respective combo box are :

- For MPs : REAL only (the choice 'DINT' will be available, but is not allowed !)
- For PVs : BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING

### 5.3.23 Field : 'Range Start'

<div>Range Start</div> <div>0.0</div>	
Full name	Measuring Range Start Value
Type of input	Numerical Value. Range : depending on sensor. Up to 3 decimals (Real)
Spreadsheet Column	O

Start of measuring Range. The value of 'Range Start' has to be less than the value of 'Range End'. Up to 3 decimals will be allowed, if 'REAL' has been selected in the 'Format' field. If 'DINT' (double Integer) has been selected in the 'Format' field, no decimals will be allowed.

This field will be visible for points with object class 'MP' and MP-type 'analog' only.

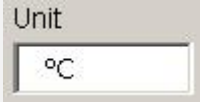
### 5.3.24 Field : 'Range End'

<div>Range End</div> <div>100.0</div>	
Full name	Measuring Range End Value
Type of input	Numerical Value. Range : depending on sensor. Up to 3 decimals (Real)
Spreadsheet Column	P

End of measuring Range. The value of 'Range End' has to be greater than the value of 'Range Start'. Up to 3 decimals will be allowed, if 'REAL' has been selected in the 'Format' field. If 'DINT' (double Integer) has been selected in the 'Format' field, no decimals will be allowed.

This field will be visible for points with object class 'MP' and MP-type 'analog' only.

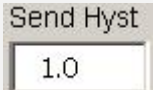
## 5.3.25 Field : 'Unit'

	
Full name	Measuring Range Unit
Type of input	Text input (ASCII Characters). Text Length : max. 5 Characters
Spreadsheet Column	Q

Technical dimension unit of the MePo (Examples: Pa, bar, Grd, RPM, kW).

This field will be visible for points with object class 'MP' and MP-type 'analog' only.

## 5.3.26 Field : 'Send Hyst'

	
Full name	Sending Hysteresis
Type of input	Numerical Value. Range : 0.5 - 2 % of absolute measuring Range
Spreadsheet Column	R

Hysteresis for initiating the transmission (sending) of analogue MePos.

The analogue values processed in the PCUs will be sent to the OPERATOR STATIONS (OS) for indication and alarming. In order to keep indications of rapidly changing values readable and (in the same time) to reduce the communication load on the bus system, PCUs will only transmit changes that are significant (i.e. that will exceed the hysteresis limit). Small changes to analogue values, which are in fact not significant for the monitoring and alarming, will not be transmitted.

This hysteresis value has to be specified as a **percentage**. This percentage refers to the total measuring range of the point. If the change of the MePo's input value is more than this percentage of the total measuring range, the value change will be transmitted.

The SendHyst field has to be filled in with the appropriate percentage value (without the '%' character, i.e. enter '1' to make this SendHyst 1% of the Measuring Range).

Too small a value for 'SendHyst' will seriously increase the communication load on the bus system. Too big a value can cause significant deviations between the real analogue value measured in the PCU and the corresponding indication / alarming in the HMI.

The recommended value for 'SendHyst' is 1% of the measuring range. Other settings are possible for special purposes. Values below 0.5% of the measuring range are likely to cause excess communication load and are not allowed. Values above 2% of the measuring range may inadvertently influence indications and alarming and should be avoided.

This field will be visible for points with object class 'MP' and MP-type 'analog' only.

**5.3.27 Field : 'Limit Hyst'**

<div>Limit Hyst</div> <div>2.0</div>	
Full name	Limit Hysteresis
Type of input	Numerical Value. Range : 1 - 5 % of absolute measuring Range
Spreadsheet Column	S

Limit Hysteresis of analogue MePos.

The Limit Hysteresis specifies a hysteresis range of limit values, within which a status change of this MePo will not appear (will be suppressed). This feature is meant to prevent frequent changes of the status of a MePo (and thus potentially frequent alarm events), if an analogue value is close to one of the limit values defined for this MePo.

This hysteresis value has to be specified as a **percentage**. This percentage refers to the total measuring range of the point. If the change of the MePo's input value is more than this percentage of the total measuring range, the status change will be transmitted.

The LimitHyst field has to be filled in with the appropriate percentage value (without the '%' character, i.e. enter '2' to make LimitHyst 2% of the Measuring Range).

Too small a value for 'LimitHyst' can potentially cause a big load of 'events' in conjunction with MePos with a significant 'jitter' on their analogue signal. Too big a value can cause the suppression of alarms that in fact have already occurred, but are hidden due to a large 'LimitHyst' value.

The recommended value for 'LimitHyst' is 2% of the measuring range. Other settings are possible for special purposes.

This field will be visible for points with object class 'MP' and MP-type 'analog' only.

**5.3.28 Field : 'CompID'**


<div>CompID</div> <div>010312</div>	
Full name	Compensation Identifier
Type of input	Text input (ASCII Characters). Text Length : max. 8 Characters
Spreadsheet Column	EL

Identifier of the compensation MePo for a thermocouple type of temperature sensor. Only relevant for objects with class 'MP' and MP-type 'analog' and sensor type 'thermocouple with compensation'.

For a precise measurement with a thermocouple type of sensor an ambient temperature compensation will be required. This additional temperature sensor is part of the compensation terminal box and will usually be a 'PT-100' sensor.

This field has to contain the reference to the MePo ID of the MePo used for this (PT-100) compensation measurement. Leave this field empty, if this is not a thermocouple-MePo or if temperature compensation is not required.

### 5.3.29 Field : 'ScalGroup'

	
Full name	Scaling Group
Type of input	Numerical Value. Range : 0 ... 99
Spreadsheet Column	T

Number of the SCALING GROUP used for his MePo. Only relevant for MePos representing an analogue value [→

8.2 Scaling Groups on page 91].

Analogue sensor input values can be transformed using an IMAC L SCALING GROUP. These SCALING GROUPS can be used for normalization, linearization, scaling and zero point adjustment of a signal. One SCALING GROUP can be used for many MePos.

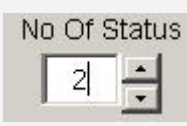
Every MePo requiring the use of a SCALING GROUP will need this field to be filled in. The number entered here is the number of the SCALING GROUP to be used for this MePo.

Leave this field empty, if normalization, linearization, scaling or zero point adjustment is not required for this MePo.

For information on how to set up SCALING GROUPS see [→

8.2 Scaling Groups on page 91].

### 5.3.30 Field : 'No Of Status'

	
Full name	Number of Status
Type of input	Numerical Value. Range : 0 ... 8 (with button dial for adjustment)
Spreadsheet Column	Z

Number of states that this MePo can have. For binary measuring points this value will be fixed to ,2' (field is not visible). Analogue MePos can have 0...8 states. Use the button dial on the right side of this field to adjust the correct number for this MePo.

For every status required, additional input is required in the 'Edit' form. Modifying the 'NoOfStatus' field will show / hide these additional fields in the 'Edit form' according to the required number of states.

This field will be visible for points with object class 'MP' and MP-type 'analog' only.

### 5.3.31 Fields : 'A Lvl'

#	A Lvl
0	15
1	12
2	0
3	

Full name	Alarm Level
Type of input	Selection using a 'combo box'. Range 0...16
Spreadsheet Column	AA,AK,AW,BI,BU,CG,CS,DE,DQ

ALARM LEVEL of every status a MePo can have. The ALARM LEVELS will determine the style in which this MePo will be represented in the IMAC L HMI (colours, blinking) and how each status will be dealt with. Each status necessarily has to be assigned to one of the defined ALARM LEVELS.


- After a status change an alarm will be generated by IMAC L, if the ALARM LEVEL of the new status is higher than the ALARM LEVEL of the prior status. That implies, that a status with the ALARM LEVEL '0' will never be able to cause an alarm.
- IMAC-L has eight ALARM LEVEL GROUPS. The (up to two) Alarm Levels within one group will be used to handle limits with a similar level of importance. Example: the levels in one group will represent the 'high' and the 'low' level warnings of a tank. The levels of the next group will represent the 'high-high' and 'low-low' level alarms of the tank.

The following table shows the Alarm levels, their priority and HMI colour representation in IMAC L.

Alarm Level (Group)	Background Colour of the Status Text in the IMAC L HMI lists	Priority
0	Green	low
1 & 2	Yellow	low
3 & 4	Orange	low
5 & 8	Red	low
9 & 10	Yellow	high
11 & 12	Orange	high
13 & 15	Red	high
16	Violet	high

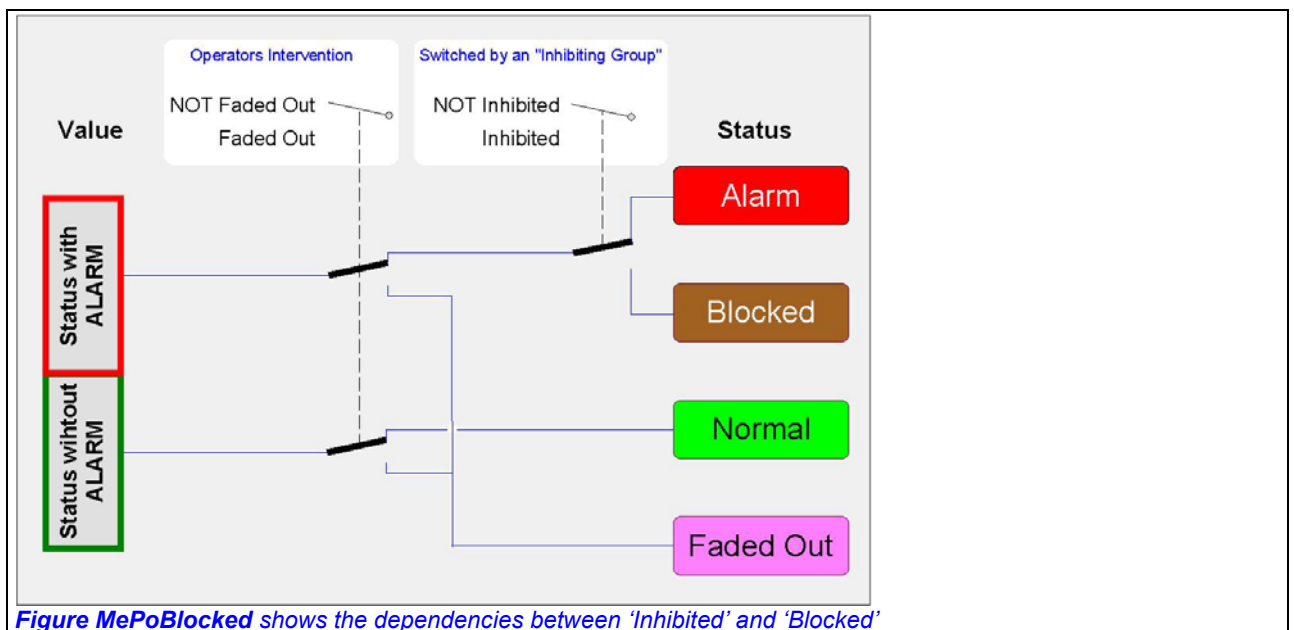


### 5.3.32 Fields : 'BLG'

	
Full name	Blocking Group
Type of input	Numerical Value. Range : 0 ... 99 (number of assigned BLOCKING GROUP)
Spreadsheet Column	AB,AL,AX,BJ,BV,CH,CT,DF,DR

In IMAC L the alarm evaluation for any ALARM LEVEL > 0 can be suppressed (inhibited). A MePo will not initiate an alarm, if this suppression is active (IMAC L calls this condition of a MePo 'INHIBITED'). A MePo that is currently in the 'inhibited' condition will be shown in the HMI with the special status 'BLOCKED', that is automatically generated by the HMI.

IMAC L will use BLOCKING GROUPS to control this function. Each BLOCKING GROUP will take its current condition (inactive / active) from a controlling MePo information (e.g. a diesel engine 'Running' feedback). If the BLOCKING GROUP is active (e.g. diesel is not running), all MePos (e.g. 'DE Oil Pressure') with references to this BLOCKING GROUP will suppress their corresponding states (e.g. 'Low') as long as the BLOCKING GROUP is active. In this example the 'Oil Pressure' MePos state 'Low' will be inhibited as long as the diesel engine is not running. In this condition the 'Oil Pressure' MePo will show the 'BLOCKED' status instead of 'Low' and no alarm will occur due to the 'low' condition. For controlling BLOCKING GROUPS see [→ 8.1 Blocking Groups on page 88].



The BLOCKING GROUP field has to contain a reference to the appropriate BLOCKING GROUP for each status that has to be blocked. For the example above, the 'Low' status of the 'DE Oil Pressure' MePo will refer to the 'DE not running' Blocking Group, while an 'Out of Range' alarm will still occur (because there is no entry in the 'BLG' field for this MePo).

If blocking is not required, this field has to be left empty.


All MePos in IMAC L that are currently in the 'BLOCKED' condition (i.e. would actually have an alarm, but this has been suppressed due to an active BLOCKING GROUP), can be viewed in the IMAC L HMI 'BLOCKED SUMMARY' (GD-Name 'XBS').

**Important Hint:** For IMAC L all MePos used in conjunction with one specific BLOCKING GROUP have to be allocated in the same PCU ! (All Blocking Group processing is done within the AbvSPU Signal Processing Unit of the respective PCU).

If it is necessary to incorporate Information from other PCUs, the required data from / to other PCUs have to be transported by a communication link. The programming of this link (send-receive) is part of the user-defined technology program. IMAC L will not generate code for this cross-communication. Data received by a PCU on a link of this kind usually have to be entered as new 'internal' MePos (i.e. data source is in a DB) in the MePo list of this PCU.

For the instructions on how to set up a BLOCKING GROUP, see [→ 8.1 Blocking Groups on page 88].


## 5.3.33 Fields : 'VAG'

	
Full name	Visual Alarm Group
Type of input	Numerical Value. Range : 0 ... 64
Spreadsheet Column	AE,AO,BA,BM,BY,CK,CW,DI,DU

Number of 'VISUAL ALARM GROUP' to which this MePo has been assigned. For every status of a MePo a VISUAL ALARM GROUP can be assigned. With this feature any status being active can be used in a PCU to actuate an (optical) indication signal. IMAC L can have up to 64 VISUAL ALARM GROUPS.

VISUAL ALARM GROUPS in IMAC L are mainly used to control the EXTENDED ALARM SYSTEM (EAS) indication for the alarm categories 'Shut Down', 'Slow Down' and 'Miscellaneous'. In this case all alarms to be indicated in the EAS have to be assigned to VA-Groups as required for the EAS system.

## 5.3.34 Fields : 'AAG'

	
Full name	Audible Alarm Group
Type of input	Numerical Value. Range : 0 ... 15
Spreadsheet Column	AD,AN,AZ,BL,BX,CJ,CV,DH,DT

AUDIBLE ALARM GROUPS ('AAG') in IMAC L can be used to control different audible signaling devices (Horns, sirens) depending on the function group or importance of an alarm. All MePos which have been assigned with one or more of their statuses to an AUDIBLE ALARM GROUP will use this AAG to indicate an alarm in one of these statuses.

The entry in this field will assign the respective status of a MePo to one of the AUDIBLE ALARM GROUPS.

Example :


- AAG 1 : Horn in Engine Control Room for important alarms.
- AAG 2 : Siren in Engine Room for less important alarms.
- AAG 2 : Buzzer in Bridge console to indicate Cargo / Bilge system alarms.

AAG can be used to specify for each status, which signaling device has to be actuated if the MePo enters this status.

Entering AAG = 0 in this field (or leaving this field empty) will cause, that no audible alarm will be caused by this status.

For evaluating AUDIBLE ALARM GROUPS and creating the required control signals see [→ 9.5 Audible Alarm and 'Horn Stop' on page 116].

## 5.3.35 Fields : 'SOG'


	
Full name	Signal Output Groups
Type of input	Numerical Value. Range : 0 ... 15
Spreadsheet Column	AF,AP,BB,BN,BZ,CL,CX,DJ,DV

Signal Output Groups (SOG) will be used in IMAC L to control external devices or systems based on status changes of a MePo [→ 9.2 Signal Output Groups on page 107]. The control can be achieved via a hardwired output in a PCU or via a data interface of any kind.

A typical application of this function is the 'Slow Down' of a propulsion diesel engine due to a limit infringement in the exhaust gas mean value deviation calculation.

With an SOG it is possible to determine for each single MePo status, which output or interface has to be actuated. The assignment of a Signal Output Group to an output or interface will be done as part of the engineering for IMAC L.

## 5.3.36 Fields : 'Status Text'

	
Full name	Status Text
Type of input	Text input (ASCII Characters). Text Length : max. 12 Characters
Spreadsheet Column	AC,AM,AY,BK,BW,CI,CU,DG,DS

This field is used to enter the respective status text for this status (e.g. HIGH, LOW, EMPTY, STOPPED). The text may contain small or capital letters as desired. Special characters and language specific characters have to be avoided. See [→3.1 Measuring Point Description Texts, page 16] for more information on the naming of statuses.

If a MePo serves for indication only (i.e. has no alarm level), the status text shall be 'Indication' and not something like 'Normal'. The text normal would falsely imply, that there are abnormal statuses possible for this MePo.

**5.3.37 Fields : 'Delay In' & 'Delay Out'**

<div>Delay In</div> <input type="text"/>	<div>Delay Out</div> <input type="text"/>
Full name	Delay In & Delay Out
Type of input	Numerical Value . Range : 1 ... 3600
Spreadsheet Column	AG&AH,AQ&AR,BC&BD,BO&BP,CA&CB,CM&CN,CY&CZ,DK&DK,DW&DX

In IMAC L every status change can be time-delayed. This delay will be separately adjustable for entering ('Delay In' ) and leaving ('Delay Out') each status. The time delay period will begin as soon as an analog value will infringe one of its limits, or if a binary signal changes its status.

If a MePo has a change between two statuses, the time delay (in & out) of the status with the higher alarm level will be active. If there is a status change between two statuses with the same alarm level, the active delays will be the ones (in & out) of the status, towards which the change is pending to occur.

The minimum time delay is one second. It is recommended to always have at least the minimum delay for MePos without any special requirements. If for a specific application a delay is undesired, leave this field empty to switch the delay completely off.

For delays > 120 seconds IMAC L will internally convert the time value (in seconds) entered in this field to a value with a resolution of 0.5 minutes. Therefore for time delays > 120 seconds the value entered in this field should be a multiple of 30 sec to avoid rounding errors.

**5.3.38 Fields : 'Limit(s)'**

<div>Limit(s) in %</div> <div>90.1</div> <div>80.2</div> <div>30.3</div>	
Full name	Limits
Type of input	Numerical Value. Range : depending on sensor. Up to 3 decimals (Real)
Spreadsheet Column	AU,BG,BS,CE,CQ,DC,DO

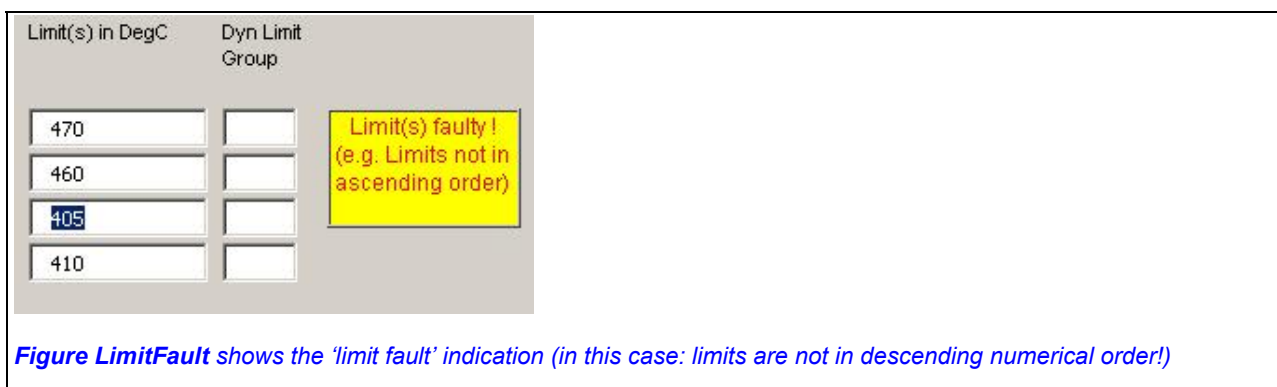
IMAC L can have up to 8 statuses and up to 7 limits for each analogue MePo. See figure [[→ Me-PoSheetStatus](#) , page 32] for a detailed description of the interrelationship between statuses and the limits assigned to these statuses.

The following basic rules apply :

- If a measured value will go through one of the limits in any direction, a status change will occur (after expiry of the respective delay time)
- An alarm will only be initiated by this, if the new status towards which the change occurs, has an ALARM LEVEL > 0 AND if the ALARM LEVEL of the new status is higher than the ALARM LEVEL of the prior status.
- Between the statuses '0' and '1' there is no limit available. Status '0' for analogue values is reserved for 'Out of Range' and thus does not require a limit value.
- The numerical values of the limits have to be assorted in descending order of numerical values : Limit1 > Limit2 > Limit3 > Limit4 > Limit5 > Limit6 > Limit7. This rule has to be satisfied, even if a dynamic limit is used (set default limit values to satisfy this rule. The defaults will be replaced by the dynamic limits as computed).

The number of decimals, that can be used for limits, is equal to the number of decimals as entered in the fields 'Range Start' / 'Range End'

If the Limits that have been defined are either invalid (not a number) or not in ascending numerical order, the following error indication will request the operator to check his input:



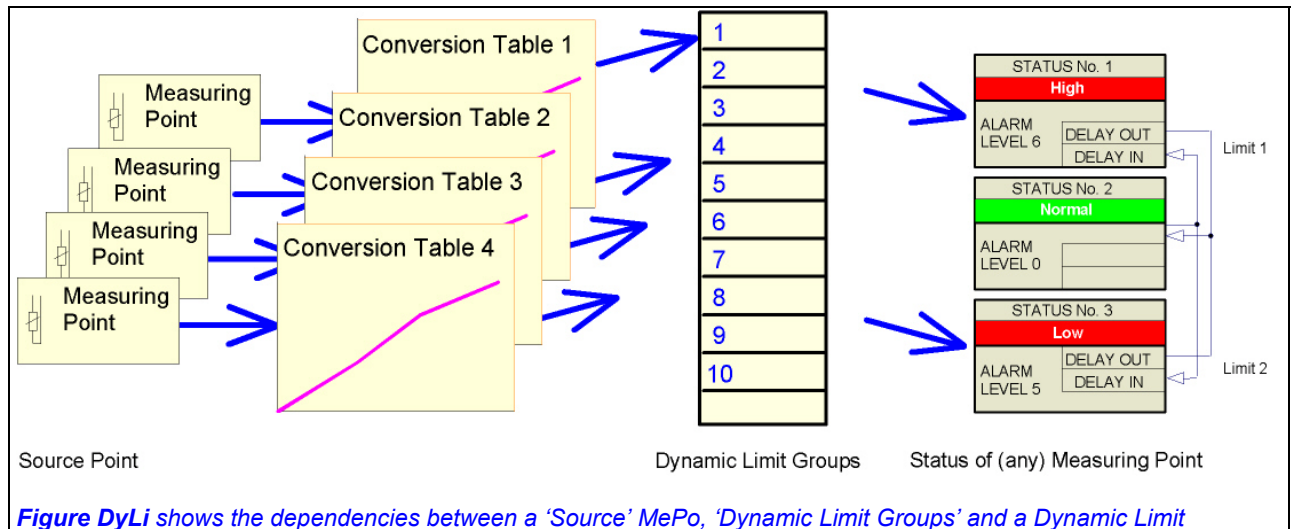
### 5.3.39 Fields : 'Dyn Limit Group'

<div> <div>Dyn Limit Group</div> <div> <input type="text"/>  <input type="text"/>  <input type="text"/> </div> </div>	
Full name	Dynamic Limit Group
Type of input	Numerical Value. Range : 0 ... 999
Spreadsheet Column	AV,BH,BT,CF,CR,DD,DP

DYNAMIC LIMIT GROUP. In IMAC L the limit values of analogue MePos can be dynamically modified depending on another MePo's data. This other MePo is called the 'Source' for dynamic limiting.

Every status can have its individual 'DYNAMIC LIMIT GROUP' assigned to it. The 'Source' MePos value will usually be processed through a CONVERSION TABLE, which is used to compute the required limit value. The output of the conversion tables will be stored in the 'DYNAMIC LIMIT GROUPS'.

The field 'Dyn. Limit Group' refers to one of these DYNAMIC LIMIT GROUPS to specify the active limit value.



If a DYNAMIC LIMIT is required, the 'Dynamic Limit Group' number of the appropriate group has to be specified for every dynamic limit. In this case the corresponding 'Limits' field for a fixed limit has to be initialized to a reasonable initial value, which is used, until IMAC L has computed the corresponding Dynamic Limit Group once (i.e. during PLC startup) !

For more information on DYNAMIC LIMIT GROUPS see [→ 8.3 Dynamic Limit Groups on page 98]

If a Dynamic limit is not required, the 'Dyn Limit Grp.' Field has to be left empty. Instead a fixed limit value has to be specified in the 'Limit(s)' field.

Example for a linear dependency between 'Source' MePo' and the computed active dynamic limit (non-linear conversion curves are possible as well) :

Source Point	Active Limit
30	2,0
40	2,5
50	3,0
60	3,5

As an example, the hydraulic pressure limit value of an aggregate can be dynamically adjusted depending on the oil temperature. Another typical example for the application of dynamic limits is the limit curve for exhaust gas mean value deviation calculations.



### 5.3.40 Field : 'Remarks'

Remarks	
<input type="text"/>	
Full name	Remarks
Type of input	Text input (ASCII Characters). Text Length : max. 20 Characters
Spreadsheet Column	Y

Remarks regarding this MePo can be entered here. This field is just for documentation purposes. Leave this field empty, if there are no remarks.

### 5.3.41 Type Indication NO / NC for Binary MePos

Binary MePos can be of three different types:

- Indication (no Alarm)
- Normally Open ('NO') (Alarm, if input signal is 1 / 'TRUE')
- Normally Closed ('NC') (Alarm, if input signal is 0 / 'FALSE')

A special indication for binary MePos will analyze the current assignment of alarm levels for binary MePos and indicate the result in the right half of the screen (red text):

**Figure MePoNC** shows a binary MePo of Type Normally Closed ( 'NC' )

**Figure MePoNO** shows a binary MePo of Type Normally Opened ( 'NO' )

### 5.3.42 'Change Type to ...' Button for binary 'NO/NC' MePos

For binary MePos in IMAC L, changing the sense of operation of the binary signal means, that all entries of the status lines '1' and '2' have to be exchanged. This would be quite some editing, just to exchange the sense of operation.

But there is a very simple way of saving this editing work. For binary MePos there is a button on the right side of the screen. This button will exchange all entries of both lines with just one click.

Normally Opened (NO)	→	'Change to NC'	→	Normally Closed (NC)
Normally Closed (NC)	→	'Change to NO'	→	Normally Opened (NO)
Indication	→	'Change to NC'	→	Indication (reversed)

This button is depicted in the 'MePoNO' and 'MePoNC' displays above in the previous section. This button changes it's text to reflect, what a click on it would do. This button is not visible for MePos not being IMAC L binary MePos.

### 5.4 Application Examples for IMAC L Measuring Points

#### Important Hints:

- always observe the figure [→ [MePoSheetStatus](#) on page 32] when assigning alarm levels to a MePo status
- The numerical values of the limits for analogue MePos have to be assorted in descending order of numerical values : Limit1 > Limit2 > Limit3 > Limit4 > Limit5 > Limit6 > Limit7 !
- The transmitter Type selected has to fit to the data source that is cited in the field 'Address'.

#### 5.4.1 Binary Measuring Point without Alarm

**Figure MePoBin** shows a typical measuring point for a binary input 'for indication purposes'

- This is a MePo for indication purposes only
- Delays (in and out) for status changes are set to one second.
- No alarms will be generated.
- This MePo will show the status 'OFF' if input E1.0 = 1 (TRUE)
- This MePo will show the status 'ON' if input E1.0 = 0 (FALSE)

### 5.4.2 Binary Measuring Point with Alarm (NO & NC Type)

Figure MePoBinAI\_NO shows a typical measuring point for a binary input with alarm (NO)

- This is a binary MePo with alarm (status 'High').
- Delays (in and out) for status changes are set to one second.
- The alarm events from 'High-alarm' will be logged and printed.
- Alarms will be signaled to AUDIBLE ALARM GROUP '1'
- This MePo will show the status 'High' if input E1.0 = 1 (TRUE)
- This MePo will show the status 'Normal' if input E1.0 = 0 (FALSE)
- → this is a '**Normally Opened (NO)**' type of alarm
- Audible Alarm Group 1 will be triggered, if this MePo enters the status 1 (High).

**Figure MePoBinAI\_NC** shows a typical measuring point for a binary input with alarm (NC)

- This is a binary MePo with alarm (status 'High').
- Delays (in and out) for status changes are set to one second.
- The alarm events from 'High-alarm' will be logged and printed.
- Alarms will be signaled to AUDIBLE ALARM GROUP '1'
- This MePo will show the status 'Normal' if input E1.1 = 1 (TRUE)
- This MePo will show the status 'High' if input E1.1 = 0 (FALSE)
- → this is a '**Normally Closed (NC)**' type of alarm
- Audible Alarm Group 1 will be triggered, if this MePo enters the status 1 (High).

### 5.4.3 Analogue Measuring Point without Limits

**Objekt**

Point ID: 010204    Obj-Class: MP    MP-Type: analog    PV-Type:    X-Type:

Description of Point (max 30 Characters): Test AI 4-20mA w/o Alarm

Station: PCU1    Section Data: 0102    TechID: 01    AIPI: NOAIPi    AIVI:    HITI:    UnderGrp:    FNoAckn:    FNoChange:    EVL: ☒    EVP: ☒

Address: PEW 128    Value (default):    InitValue: ☐    Transmitter: AI 4..20mA    Terminal: PCU1.A217.X4    Format: REAL

Range Start: 0    Range End: 100    Unit: °C    Send Hyst: 1    Limit Hyst: 1    Compld:    ScalGroup:    No Of Status: 1

#	A Lvl	BLG	VAG	AAG	SOG	Status Text	Delay In	Delay Out	Limit(s) in °C	Dyn Limit Group
0	15			1		Out Of Range	1	1		
1	0					Normal	1	1		
2										
3										
4										
5										
6										
7										
8										

Remarks:

Last Change: 13.09.2004 17:06:29    Excel Row No: 32

User: 57125400@DE2ZEV6D

Buttons: Previous, Move UP, Insert bef., Discard Chg., Save, Print, Goto Row, Next, Move DOWN, Insert after, Delete, Save as, Close

**Figure MePoAna** shows a typical measuring point for an analogue value (no limits, Out of Range Alarm only)

- This is an analogue MePo without limits.
- Delays (in and out) for status changes are set to one second.
- Normal value changes will not cause an alarm.
- Measured values 'Out-of-Range' will cause an alarm, which will be logged and printed.
- Alarms will be signaled to AUDIBLE ALARM GROUP '1'



### 5.4.4 Analogue Measuring Point with Limits

The screenshot shows the 'Objekt' configuration window for an analogue measuring point. The configuration includes the following fields and values:

- Point ID:** 010205
- Obj-Class:** MP
- MP-Type:** analog
- Description of Point (max 30 Characters):** Test AI 10V with 4 Alarms
- Station:** PCU1
- Section Data:** 0102
- TechID:** 01
- AIPI:** NOAIPI
- AI/VI:** AI +/-0..10V
- Terminal:** PCU1.A219.X4
- Format:** REAL
- Range Start:** 0
- Range End:** 100
- Unit:** %
- Send Hyst:** 1
- Limit Hyst:** 1
- Compld:** (empty)
- ScalGroup:** (empty)
- No Of Status:** 4

The alarm limits table is configured as follows:

#	A Lvl	BLG	VAG	AAG	SOG	Status Text	Delay In	Delay Out	Limit(s) in %	Dyn Limit Group
0	15			1		Out Of Range	1	1		
1	13			1		Very High	2	2	90.1	
2	12			1		High	1	1	80.2	
3	0					Normal	1	1	30.3	
4	2			1		Low	1	1		

At the bottom, there are fields for 'Last Change' (13.09.2004 17:08:25), 'Excel Row No' (31), and a 'User' field (57125400@DE2ZE6D). Navigation buttons include Previous, Move UP, Insert bef., Discard Chg., Save, Print, Goto Row, Next, Move DOWN, Insert after, Delete, Save as, and Close.

**Figure MePoAnaAI** shows a typical measuring point for an analogue value (3 limits + Out of Range)

- This is an analogue MePo with three limits.
- Delays (in and out) for status changes are set to one second. (two seconds for 'very high').
- Values below 30.3 °C will cause an alarm 'Low'
- Values between 30.3 °C and 80.2 °C will not cause an alarm ('Normal').
- Values above 80.2 and below 90.1 °C will cause an alarm 'High'
- Values above 90.1 °C will cause an alarm 'Very High'
- Measured values 'Out-of-Range' will cause an appropriate alarm.
- Alarms will be logged and printed.
- All Alarms will be signaled to AUDIBLE ALARM GROUP '1'

## 6 Simatic S7 Configuration

### 6.1 Simatic S7 Hardware Configuration

The Simatic Hardware configuration has to be edited according to Simatic rules. The hardware modules being used in a project have to be entered to the Simatic Manager hardware configuration in precisely the configuration as it will be used in the project.

The address assignment has to be performed according to standard Simatic procedures.

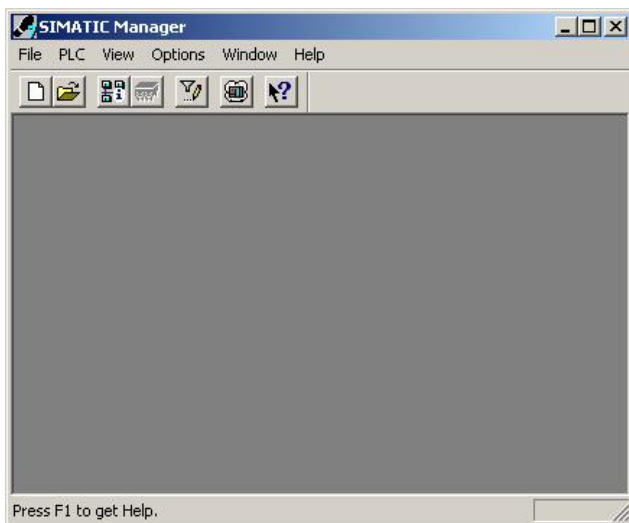
It is recommended to design this hardware configuration prior to beginning the editing of MePo data, because the addresses used for the peripherals will be required for the MePo design.

To edit the Simatic S7 hardware configuration proceed as follows :

- Double-click on the 'Simatic Manager' icon on your desktop:

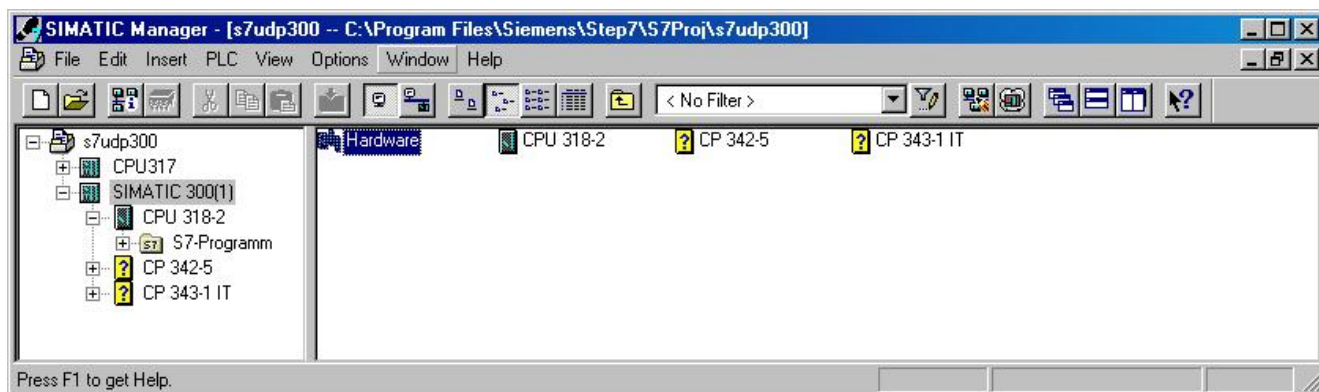


- The Simatic Manager window will open :

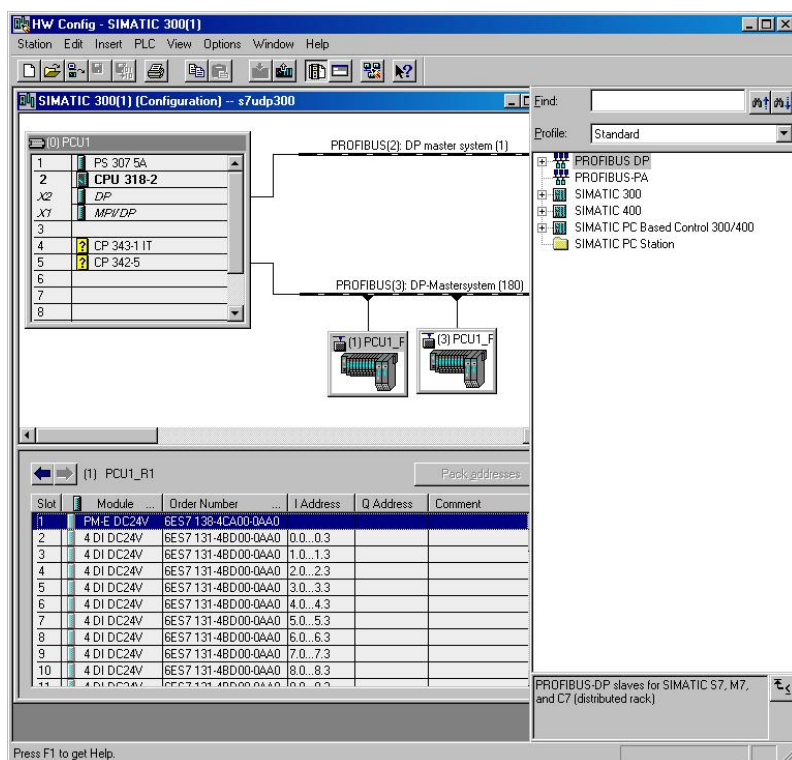


- Click File → Open and select the IMAC L Template file (as it has been unpacked during the installation sequence)
- Open the Program tree and double-click on Simatic 300 → Hardware





- (the Simatic Hardware-Configuration tool will open)
- Edit the hardware configuration to fit your current hardware. Be careful not to modify the default settings and connections of the 'CP343-IT' communications processor. These settings are required to establish the UDP communication links for IMAC L. Modifying these settings or simply placing a CP without the settings and connections as delivered with the IMAC L template project will render the bus communication on the Ethernet bus inoperative.
- Observe all standard Simatic rules for configuring and parameterizing hardware components (no additional / special IMAC L requirements for I/O).

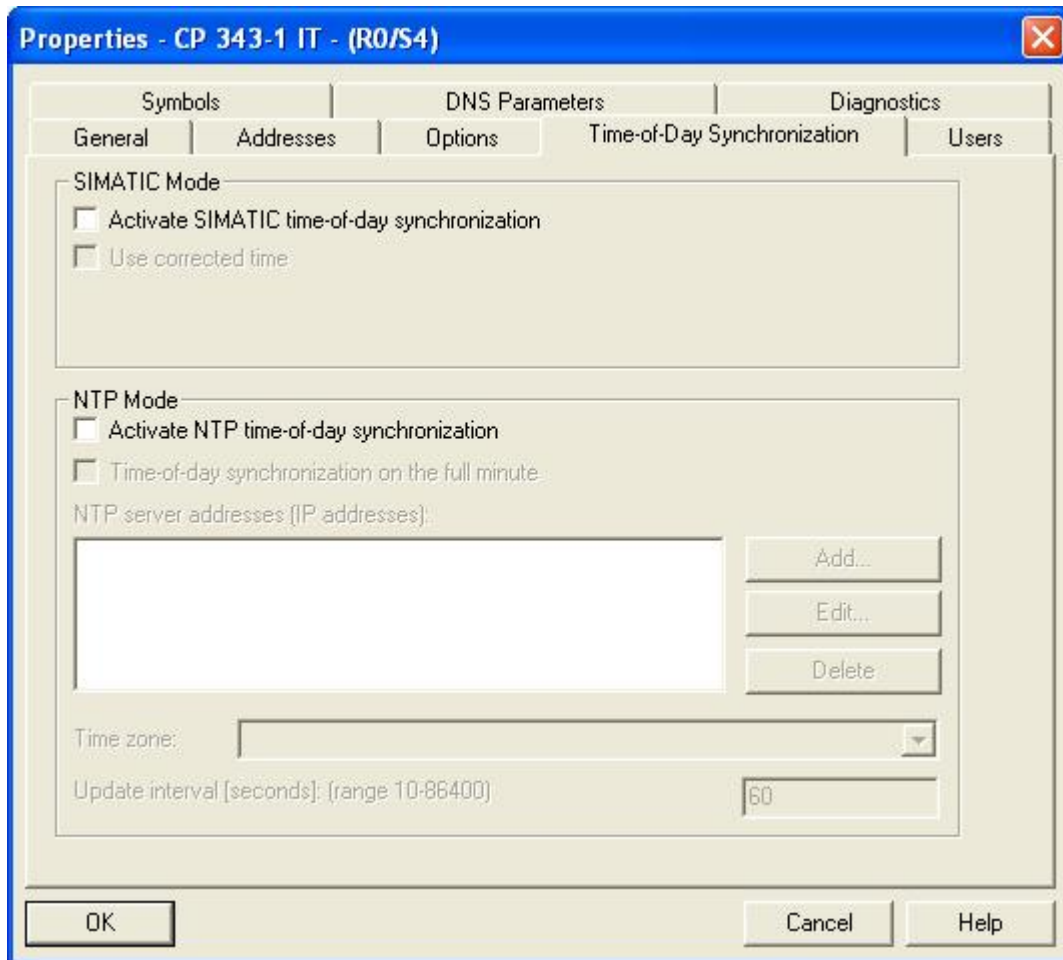


After editing this hardware configuration, it has to be downloaded to the Simatic S7 in order to become effective. The Simatic S7 will be stopped in this sequence, if it is currently running.

If you are using Simatic Net 'Softnet' to access the S7 via the Ethernet bus, the first download of the hardware configuration has to be performed using an MPI connection to the PLC. The Ethernet connection via 'Softnet' will only be available, after correctly initializing the hardware configuration (i.e. the configuration of the CP343-IT processor has been performed once).

The 'Simatic Time-of-day Synchronization' has to be switched OFF for every Ethernet CP you use in an IMAC L project. To switch it off:

- double-click on the Ethernet CP in the hardware configuration
- Click on Tab 'Time-of-Day Synchronization'
- Uncheck the checkbox 'Activate SIMATIC Time-of-Day Synchronization'




IMAC L uses it's own (internal) time synchronization protocol. If you have only one CP with the 'SIMATIC Time-of-Day Synchronization' switched on, these two algorithms will start to battle in case of time setting or clock deviations. This will NOT work.

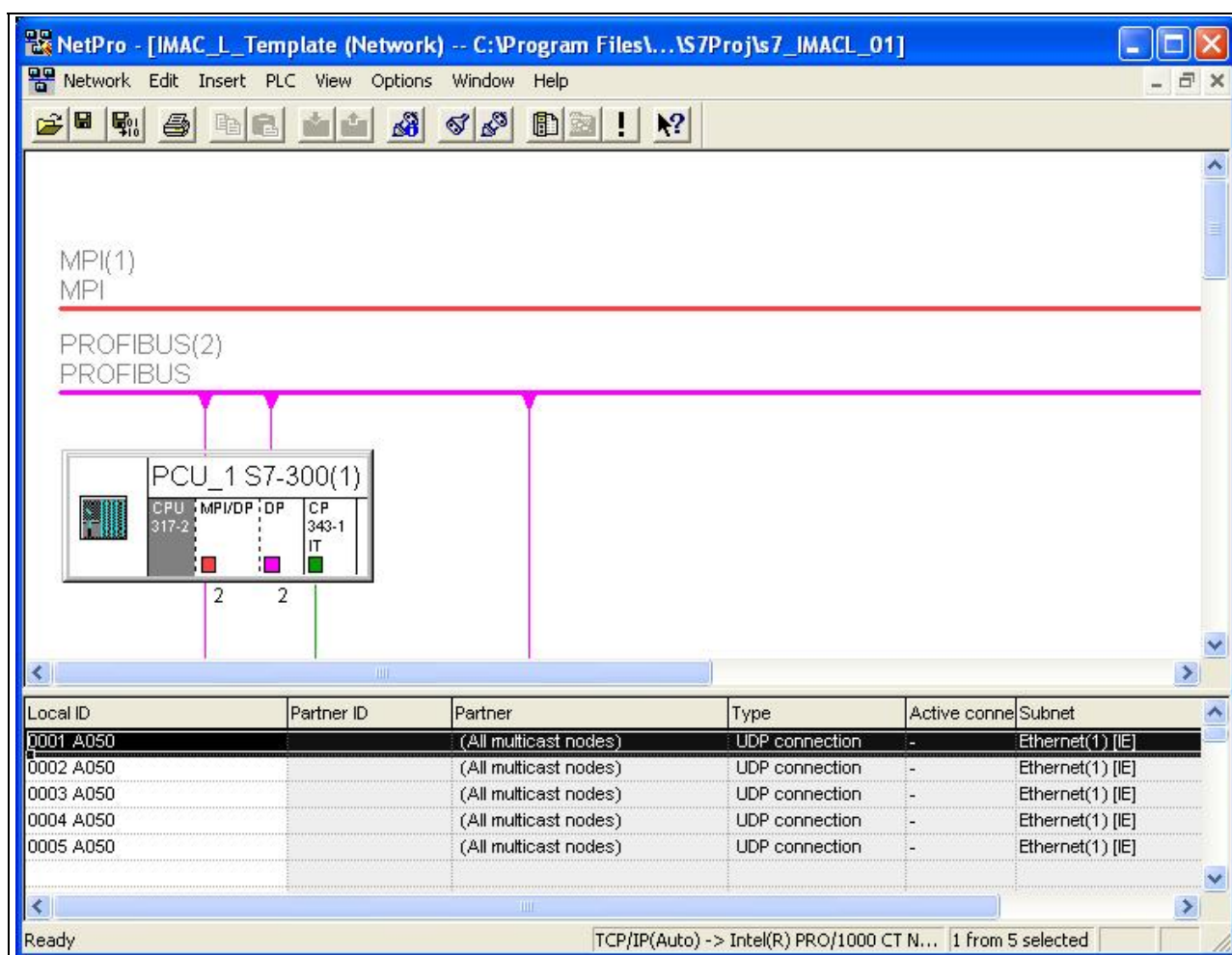
### 6.2 Simatic S7 'NetPro' Configuration

IMAC L requires a set of five UDP connections in each IMAC L PCU. For the predefined IMAC L template project contained in your 'IMAC L CD' these connections are already predefined and ready to use. There are situations, where it is necessary to check (and add if missing) modify or add these connections:

- If you have created a new IMAC PCU in your project (new PLC in hardware configuration)
- If you have replaced a CP343 processor in your hardware configuration
- If you experience partial or total malfunction in Ethernet communication

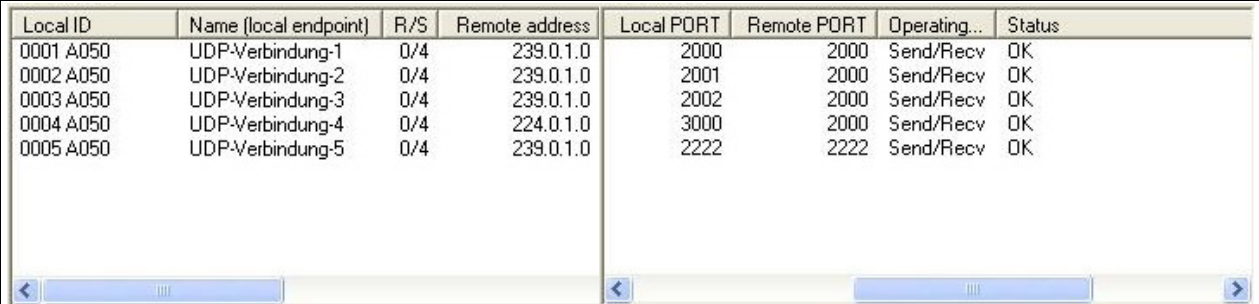
In such cases, please check the required UDP connections of all PCUs in your project according to the following procedure:

- In the Simatic Manager open your IMAC L Simatic project to be checked.
- Click on the 'NetPro' button : 
- (the Simatic NetPro-Configuration tool will open)
- Click on the CPU of the PCU to be checked
- (a list of connections will be displayed in the lower part of NetPro)
- Check, if there are five UDP type connections in this window:



**Figure NetProCon1** shows the required five UDP connections required in each IMAC L PCU

- Double-Click on any of the five connections
- (a popup window 'Properties – UDP connection' will appear)
- Click on Tab 'Overview'
- Precisely check the settings of the five UDP connections:



Local ID	Name (local endpoint)	R/S	Remote address	Local PORT	Remote PORT	Operating...	Status
0001 A050	UDP-Verbindung-1	0/4	239.0.1.0	2000	2000	Send/Recv	OK
0002 A050	UDP-Verbindung-2	0/4	239.0.1.0	2001	2000	Send/Recv	OK
0003 A050	UDP-Verbindung-3	0/4	239.0.1.0	2002	2000	Send/Recv	OK
0004 A050	UDP-Verbindung-4	0/4	224.0.1.0	3000	2000	Send/Recv	OK
0005 A050	UDP-Verbindung-5	0/4	239.0.1.0	2222	2222	Send/Recv	OK

*Figure NetProCon2 shows the connection details for the five IMAC L UDP connections*

If there are either connections missing or if these settings are not correct, this necessarily has to be fixed to make IMAC L work.

If you found wrong settings in a connection, please modify settings as follows:

- In NetPro Click on the CPU of the PCU to be modified
- Double-Click on the connection to be modified (in the lower part of NetPro)
- Edit the connection properties in the tabs 'General Information' and 'Addresses' to fit the required settings in figure 'NetProCon2' (above in this section)
- Click on 'OK' to accept changes
- Save, compile & load the modified hardware configuration

If UDP connection(s) are completely missing, please add it as follows:

- In NetPro Click on the CPU of the PCU to be modified
- Click on the topmost empty row in the list of connections (in the lower part of NetPro)
- Right-Click on 'Insert New Connection'
- In the upper window of the dialogue click on 'All Multicast Stations'
- In the lower window select Type = 'UDP Connection'
- Make sure that the CheckMark 'Display Properties before inserting' is NOT checked
- Click on 'Apply'
- Click on 'OK' in the upcoming warning windows
- Click on 'Close'
- Double-Click on the newly inserted connection
- Edit the connection properties in the tabs 'General Information' and 'Addresses' to fit the required settings in figure 'NetProCon2' (above in this section)
- Click on 'OK' to accept changes
- Save, compile & load the modified hardware configuration

### **Remark:**

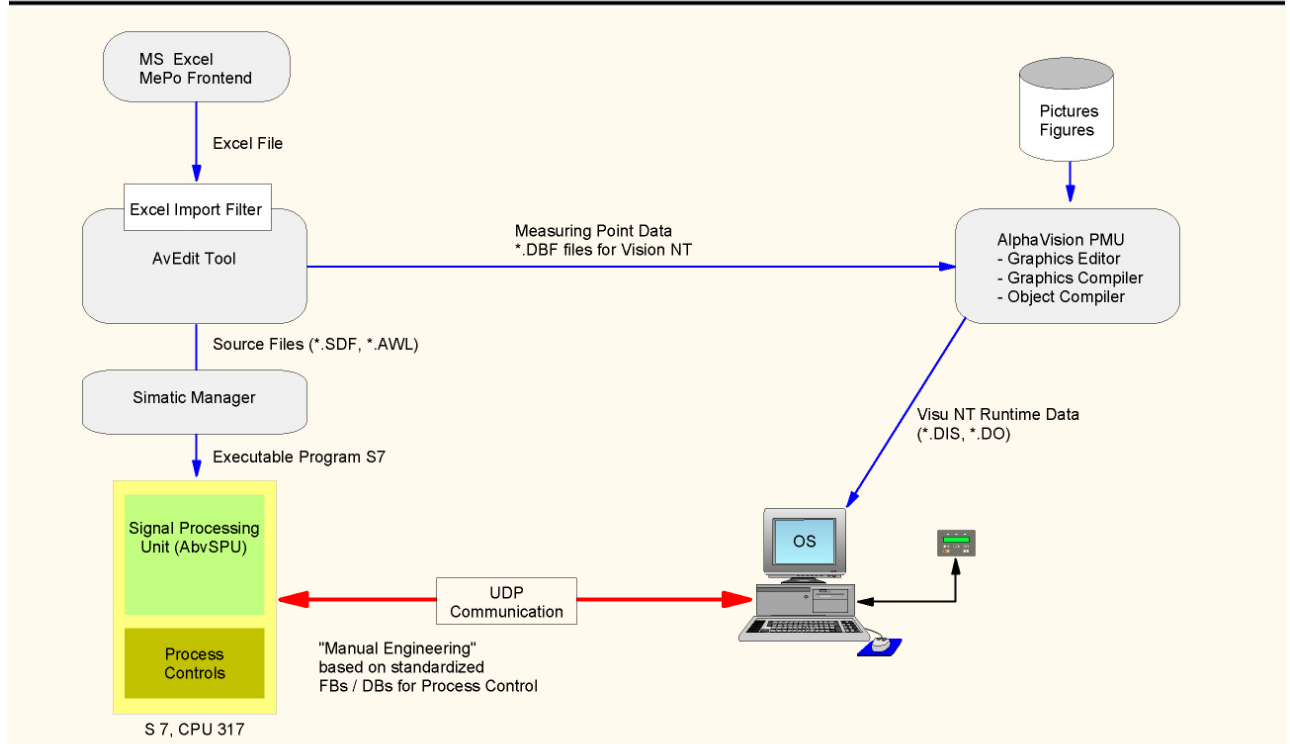
If there are communication links in your project, which have been engineered using Simatic standard communication functions, some or all of these other links may show up in the NetPro list as well (in addition to those five required for IMAC L). In this case you have to carefully identify the right IMAC L UDP connections for checking / modifying.

## 7 Generating Simatic S7-Code and Vision NT Runtime Data

The following overview shall help to understand the sequence of steps necessary to generate an IMAC L system starting from the MS Excel spreadsheet containing the MePo data.

### SISHIP IMAC L

### Functional Overview, Principle



Copyright Siemens AG 2004, all Right reserved, I&S IP PEP SN | Rel 040928 Ho

\\bz501a\www001\siemens\net\2002\projekte\IMAC\_55U\Aut\1=LeitSys\L14=Para&Struktur\ParaDescription\Stapshots\55L\_Engineering\_Overview\_040928.dcf

The required handling of all tools referenced in this overview picture will be described in detail in the following chapters :

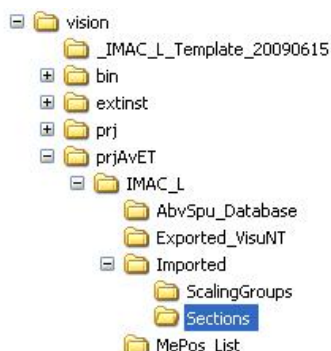
### 7.1 Deleting of old AvET data from previous Excel-Importing

If you have previously generated S7-Code, you will have to manually delete old data resulting from previous importing as follows :

#### 7.1.1 Deleting of Sections

Delete all XML-files contained in the following directory, but preserve the empty directory :

- 'C:\vision\prjAvET\IMAC\_L\Imported\Sections'





For Importing from Excel this directory has to be empty. The files contained in this directory will be automatically generated during Excel import.

### 7.1.2 Deleting of obsolete Measuring Points

In IMAC L Measuring Points (MPs) are stored in the '\_XlsMP.obj.xml' list. This list will be empty, for the first Excel import of a project.

If you remove a MePo from your Excel list, that has been previously imported to AvEt, you will have to manually remove this MP from the AvEt '\_XlsPV.obj.xml' list in order to remove this MePo completely.

Excel importing will add / modify MePos in AvET, but not delete them.

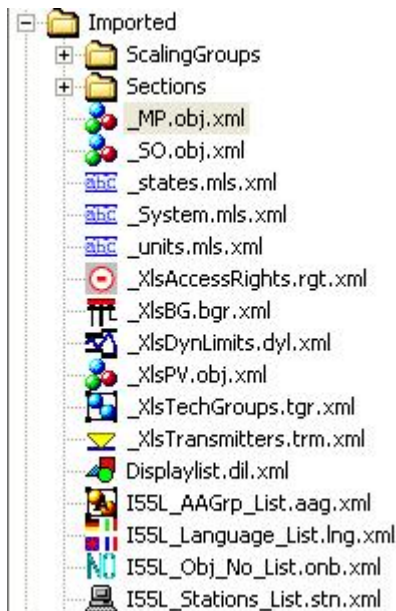
You may either remove all entries of this list (and subsequently re-import all of them from Excel again) or you may select and delete single entries as required. Observe, that inserting / deleting may cause a new assignment of object numbers in AvEt (important for control programs accessing MePo data by referencing to the object number).

To remove all or selected MPs from the '\_XlsMP.obj.xml' list :

- Double-Click on the AvET (AlphaVision Engineering Tool) Icon on your Desktop :
- Click on 'File → Open'
- Point to file C:\vision\prjAvET\IMAC\_L\I55L\_Template.prj.xml
- Click on 'Open'



- Double-Click on the 'Imported' Folder





- Double-Click on the '\_MP.obj.xml' list.

In the middle window a list of all MePos will be displayed:

Name	Description	Station	object number	help-ID
E...	Enter text here	E...	Enter te...	E...
010101	Test Point Binary (Ind.)	PCU1 ()	1	010102
010102	Test Point Binary (NC)	PCU1 ()	2	010102
010103	Test Point Binary (NO)	PCU1 ()	3	NoHiTi
010104	Sample Point Analog (Ind.)	PCU1 ()	1500	NoHiTi
010105	Sample Point Analog (2 Limits)	PCU1 ()	1501	NoHiTi
010106	Test Blocking from 010101-cl	PCU1 ()	4	NoHiTi
010107	Test Blocking from 010102-...	PCU1 ()	5	NoHiTi
010108	Inp. UndefGrp 1->010102	PCU1 ()	6	NoHiTi
010109	Inp. UndefGrp 2 (inv) ->01...	PCU1 ()	7	NoHiTi
010115	Test Scaling 01	PCU1 ()	1502	NoHiTi
010116	Test Scaling 02	PCU1 ()	1503	NoHiTi
010117	Test Scaling 03	PCU1 ()	1504	NoHiTi
010118	Test Scaling 04	PCU1 ()	1505	NoHiTi
010119	Test Scaling 05	PCU1 ()	1506	NoHiTi
010120	Test Scaling 06	PCU1 ()	1507	NoHiTi
010121	Test Scaling 07	PCU1 ()	1508	NoHiTi
010122	Test Scaling 08	PCU1 ()	1509	NoHiTi
010123	Test Scaling 09	PCU1 ()	1510	NoHiTi
010124	Test Scaling 10	PCU1 ()	1511	NoHiTi
010125	Test Scaling 11	PCU1 ()	1512	NoHiTi
010126	Test Scaling 12	PCU1 ()	1513	NoHiTi
010127	Sample Point Temp (Ind.)	PCU1 ()	1514	NoHiTi
020101	Test Point Binary Section 2	PCU1 ()	8	NoHiTi
030122	MF Operation Mode	PCU1 ()	9	NoHiTi

To delete objects :

- Select single or all objects in this list and press 'DEL'
- Click on the 'Apply' button (  ) in the right window to save the changes to this list.
- Click on the 'Save' button (  ) of AvET.

### 7.1.3 Deleting of obsolete Process Variables

In IMAC L Process Variables (PVs) are stored in the '\_XlsPV.obj.xml' list. To view a list of PVs :

- Double-Click on the '\_XlsPV.obj.xml' list.



In the middle window a list of all PVs will be displayed. This list usually will contain a list of standard PVs from the IMAC L template project. These PVs will be present in AvET only, but not in the



Excel MePo-frontend. Do not delete or modify these standard PVs. You may add your own PVs by just adding them to your MePo list as described above. If you remove a 'custom' PV from your Excel list, that has been previously imported to AvEt, you will have to manually remove this PV from the AvEt '\_XlsPV.obj.xml' list in order to remove this PV completely.

Excel importing will add / modify PVs in AvET, but not delete them.

To delete a PV :

- Select all objects in this list and press 'DEL'
- Click on the 'Apply' button () in the right window to save the changes to this list.
- Click on the 'Save' button () of AvET.

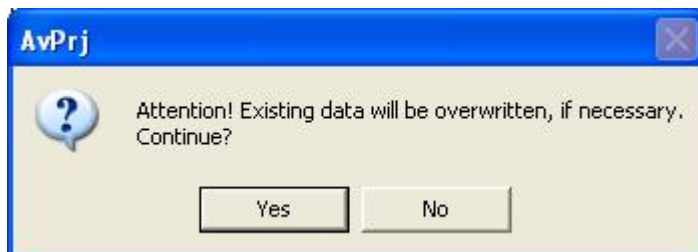
If the standard PVs from the IMAC L template project have been inadvertently deleted or modified, it is possible to copy the '\_XlsPV.obj.xml' list from the IMAC L template project back to your project (close AvET before doing so). This will remove all of your 'custom' PVs and get back all required standard PVs. You may then add your 'custom' PVs again by simply importing your MePo list once again.

## 7.2 Importing MePo data from Excel

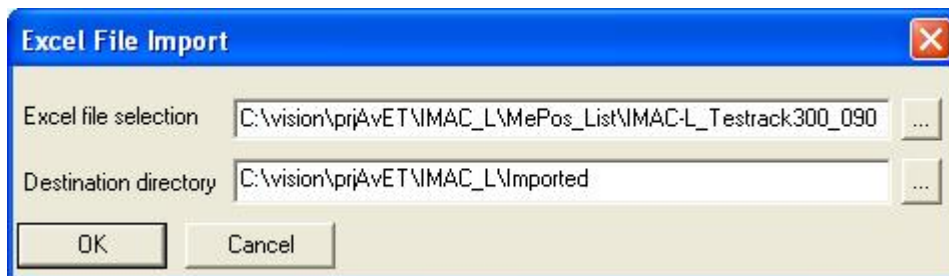
- Click on the 'Excel Import' Button :



- A dialogue window will hint, that existing MePo data will be overwritten :

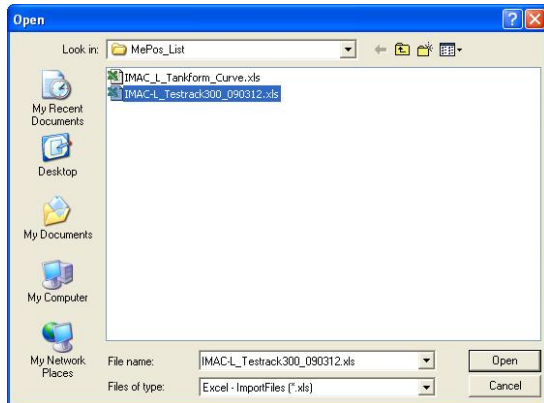


- Click on the 'Yes' button to start importing Excel data
- A dialogue window will ask, which excel file is to be imported. The second input is for the destination directory in IMAC L:

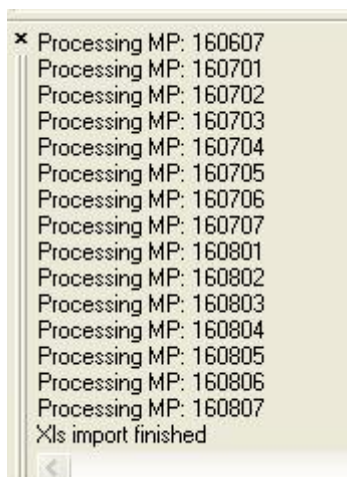


- In the first input line point to the Excel file to be imported (default folder for IMAC L is 'c:\vision\prjAvET\IMAC\_L\MePo\_List\').

- You may optionally use the 'browse' button on the right side to select an Excel file using a file browser window. Point to a file and click on 'Open'.



- In the second input line point to the import destination Directory 'C:\vision\prjAvET\IMAC\_L\Imported' (browsing is available as well).
- Click on the 'OK' button to start importing Excel data
- The result will be displayed as follows (finish message is 'Import beendet') :



All MePos and PVs contained in the Excel sheet will be imported. For larger counts of MePos this may take a while to complete.

### 7.3 Automatic Assigning of Object Numbers

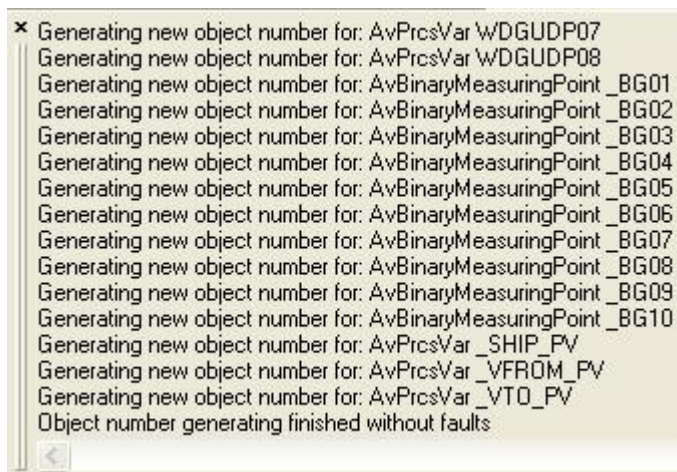
- Click on the 'Assign Object Numbers' Button :



- AvET will prompt you for the way object numbers are treated:



- Always choose '**New Generation of all object numbers**' (other options reserved for special use).
- Click on 'OK'
- AvET will automatically assign object numbers to MePos and PVs :



### 7.4 Generating the AbvSPU Code for Simatic S7

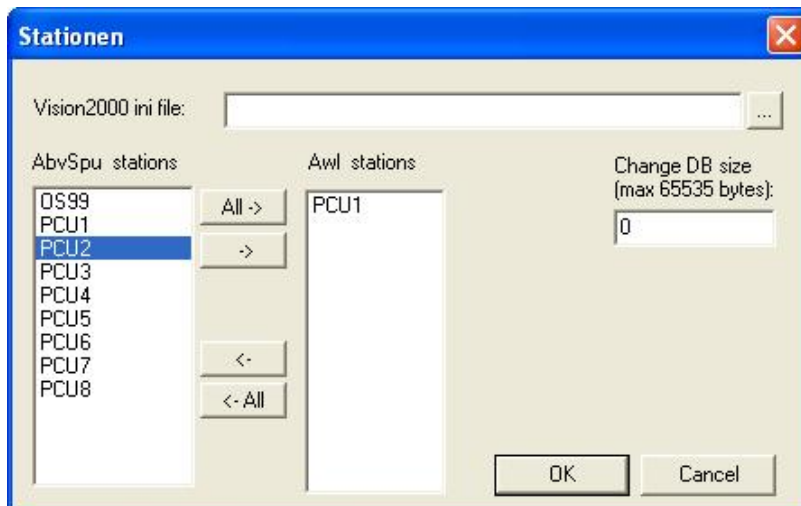
- Click on the 'Generate AbvSPU' Button :



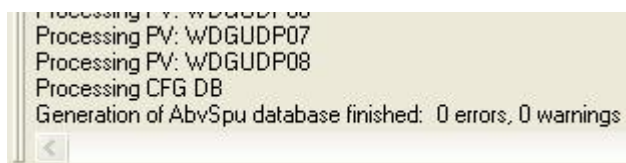
- AvET will prompt you for the way object numbers are treated:



- Always choose '**New Generation of all object numbers**' (other options reserved for special use).
- Click on 'OK'
- Select a PCUs, for which IMAC L code shall be generated by clicking on it. Use the 'right arrow' button ( ' -> ' ) to add this PCU to the list ('Awl stations') of PCUs to be generated. Multiple PCUs can be generated in one step (add to list one after the other).



- Click on 'OK' to start the exporting procedure
- The result will be displayed as follows :



Warnings and error messages from the code generator are usually caused by faulty entries in the Excel MePo Front-end and will end up in single MePos not-working.

## 7.5 Generating the VisuNT database files for AlphaVision

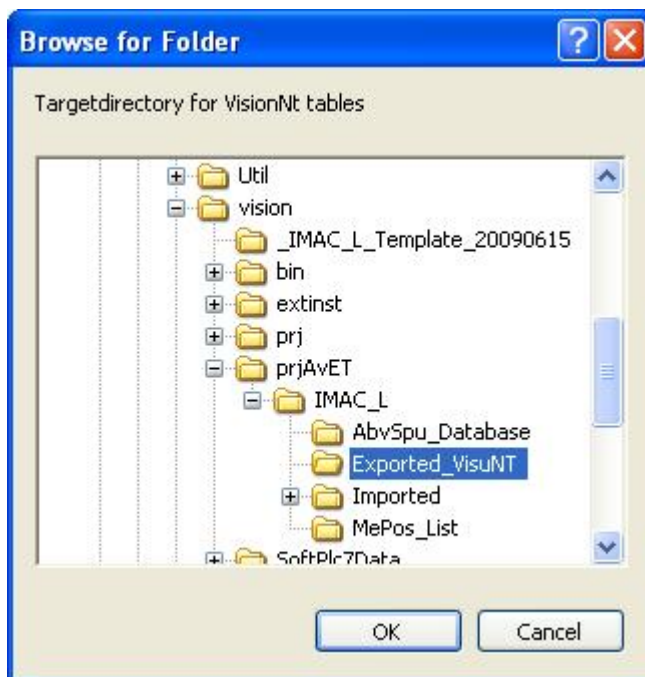
- Click on the 'Generate VisuNT' Button :



- AvET will prompt you for the way object numbers are treated:

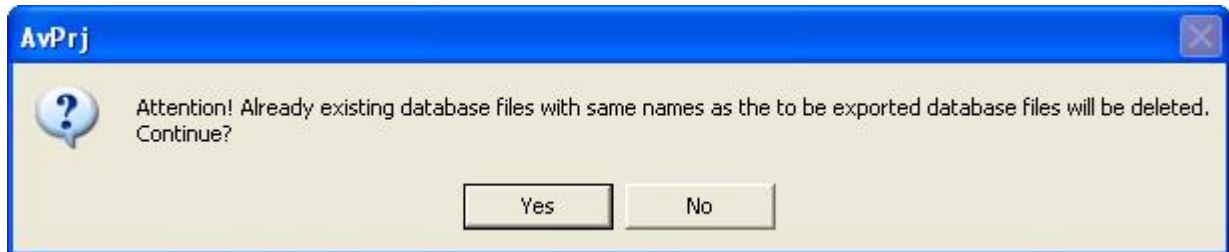


- Always choose '**New Generation of all object numbers**' (other options reserved for special use).
- Click on 'OK'
- Select the target directory for export. Always use the directory as defined by IMAC L:  
**'C:\vision\prjAvET\IMAC\_L\Exported\_VisuNT'**.  
(otherwise the IMAC L 'copy' batch files will not work)



- Click on 'OK'

- If there are already old DBF-files existing in this directory, the following dialogue will indicate, that these files are to be overwritten now:



- Select 'Yes'
- The result will be displayed as follows :



Close 'AvET' now.

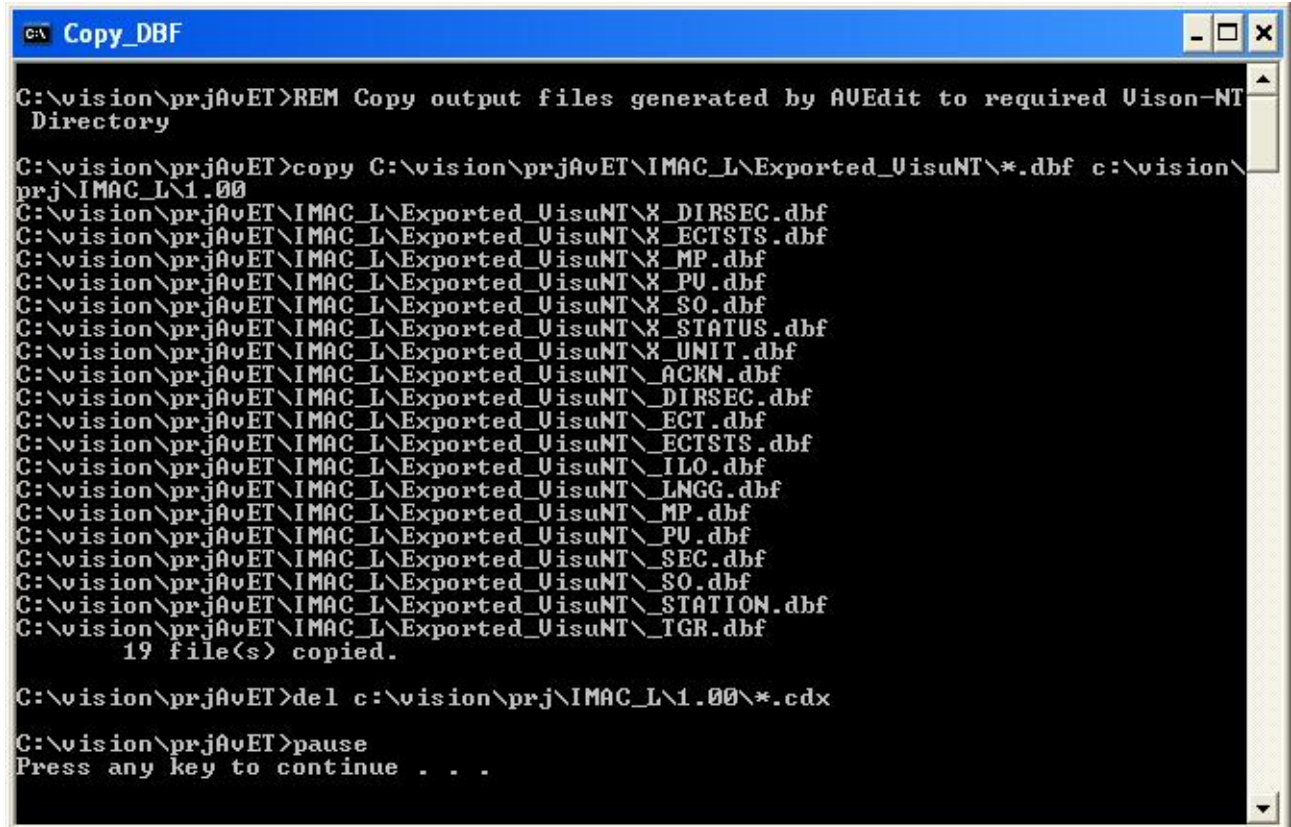


## 7.6 Copying of DBF-Files for Vision-NT

- Double-Click on the Desktop Icon 'Copy\_DBF' :



- A shell window will open indicating the correct copying of all generated DBF-files :



```
C:\vision\prjAvET>REM Copy output files generated by AVEEdit to required Vison-NT
Directory

C:\vision\prjAvET>copy C:\vision\prjAvET\IMAC_L\Exported_VisuNT\*.dbf c:\vision\
prj\IMAC_L\1.00
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\X_DIRSEC.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\X_ECTSTS.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\X_MP.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\X_PU.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\X_SO.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\X_STATUS.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\X_UNIT.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\ACKN.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\DIRSEC.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\ECT.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\ECTSTS.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\ILO.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\LMGG.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\MP.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\PU.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\SEC.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\SO.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\STATION.dbf
C:\vision\prjAvET\IMAC_L\Exported_VisuNT\TGR.dbf
        19 file(s) copied.

C:\vision\prjAvET>del c:\vision\prj\IMAC_L\1.00\*.cdx

C:\vision\prjAvET>pause
Press any key to continue . . .
```

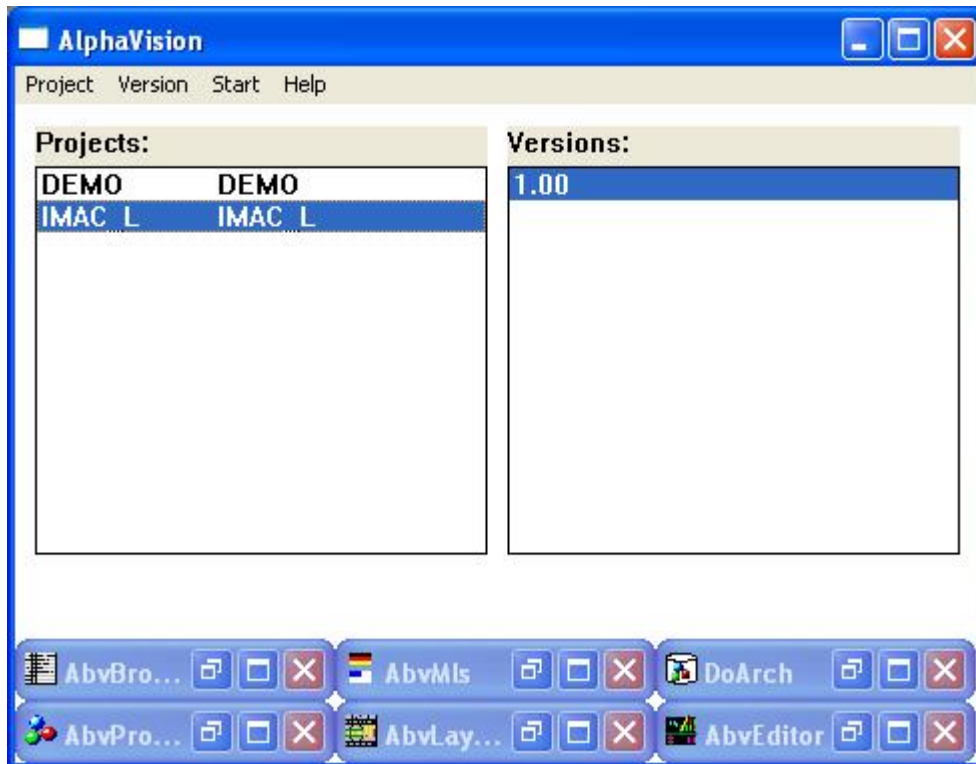


### 7.7 Compiling of AlphaVision Graphics Pictures

- Double-Click on the Desktop Icon 'Pmu' :



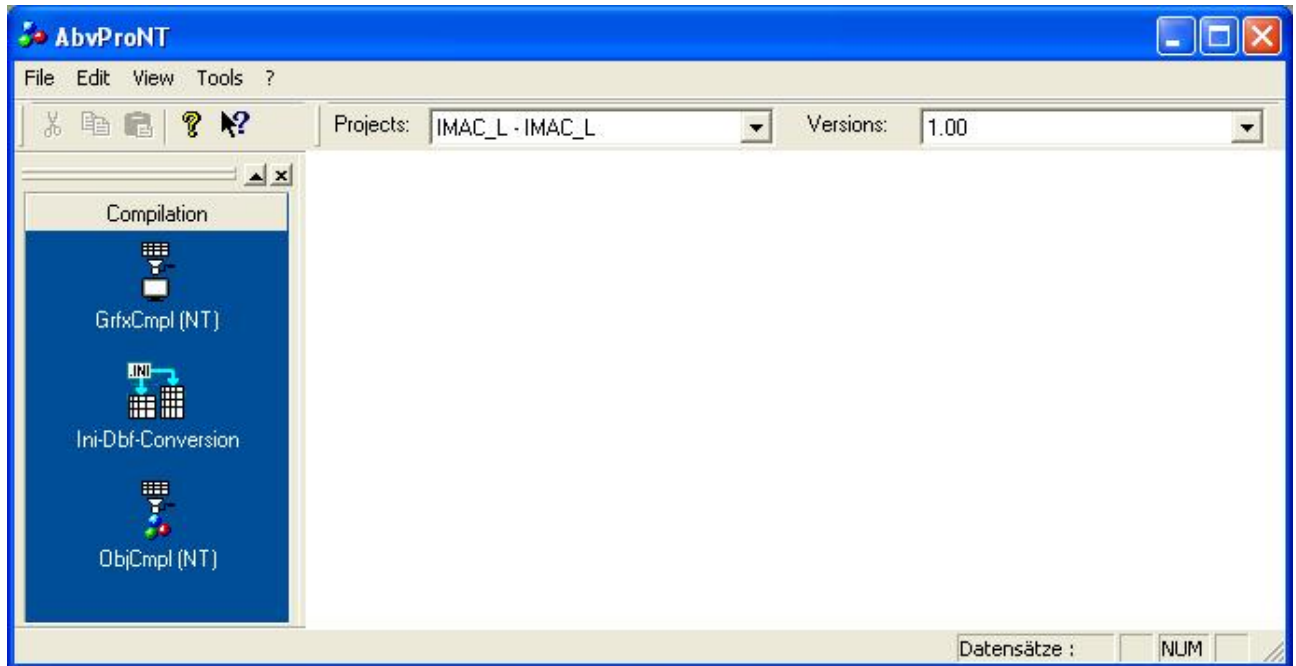
- Select the Project 'IMAC\_L' and the Version '1.00' by clicking on them (these should exist, if the OS / Engineering PC has been setup according to instructions) :



- Double-Click on the 'AbvProNT' Tool :



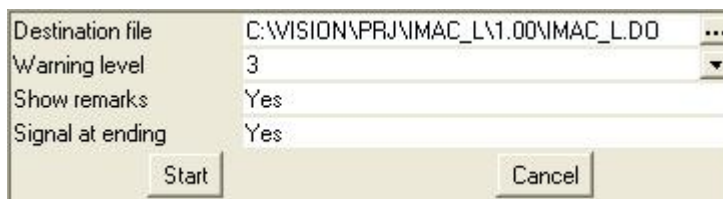
- The AbvProNT Window will open :



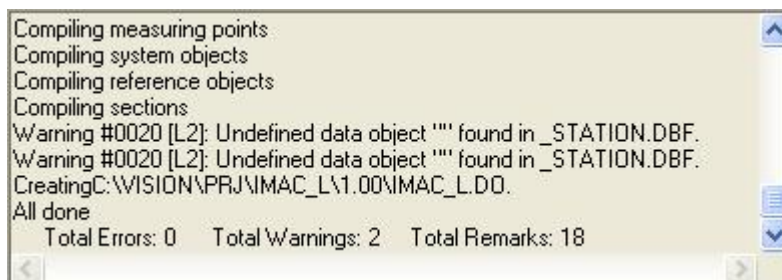
- Double-Click on the 'ObjCmpl(NT)' Icon :



- Enter the destination file and directory as shown below
- Click on 'Start'



- The results will be displayed as follows (the shown 2 warnings are normal and have to be ignored, but other warnings or even errors have to be tracked. Remarks can be ignored.) :



- Double-Click on the 'GrfxCmpl(NT)' Icon :



- Enter the destination file and directory as shown below
- Click on 'Start'

Destination file	C:\VISION\PRJ\IMAC_L\1.00\IMAC_L.DIS	...
Alarm level	3	▼
Show remarks	Yes	
Signal by ending	Yes	
<input type="button" value="Start"/> <input type="button" value="Cancel"/>		

- The results will be displayed as follows (any warnings or errors have to be tracked) :

XSOP1	▲
XUS	
XUSP	
XVS	
XVSP	
XVSP1	
Creating C:\VISION\PRJ\IMAC_L\1.00\IMAC_L.DIS.	
All done	
Total errors: 0    Total warnings: 0    Total remarks: 0	
◀ ▶	

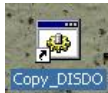
**Hint :** Depending on the number of graphical objects in a project this compilation run may take a serious amount of time (2 – 10 min as a guess). Do not interrupt this procedure, even if PMU / Ab-vProNT does not seem to react any more !

## 7.8 Copying of Visu Runtime Files (\*.DIS / \*.DO)

### Attention :

Before copying new DIS / DO files to the OSs you have to make sure, that all AlphaVision applications on all OSs have been terminated correctly. Not observing this rule may render running OSs useless or can even cause crashes.

- Double-Click on the Desktop Icon 'Copy\_DISDO' :



- A shell window will open indicating the correct copying of all Runtime-files :

```

C:\vision\prj\IMAC_L\1.00\IMAC_L.DIS
1 file(s) copied.

C:\vision\prj\AVET>copy c:\vision\prj\IMAC_L\1.00\*.do x:\vision\bin
c:\vision\prj\IMAC_L\1.00\IMAC_L.DO
1 file(s) copied.

C:\vision\prj\AVET>copy c:\vision\prj\IMAC_L\1.00\*.dis y:\vision\bin
c:\vision\prj\IMAC_L\1.00\IMAC_L.DIS
1 file(s) copied.

C:\vision\prj\AVET>copy c:\vision\prj\IMAC_L\1.00\*.do y:\vision\bin
c:\vision\prj\IMAC_L\1.00\IMAC_L.DO
1 file(s) copied.

C:\vision\prj\AVET>net use x: /DELETE
x: was deleted successfully.

C:\vision\prj\AVET>net use y: /DELETE
y: was deleted successfully.

C:\vision\prj\AVET>pause
Press any key to continue . . .
  
```

### Hints :

If you have more or less OSs, the Copy\_DISDO batch file has to be modified to distribute the required runtime files (across network shares) to all Operator Stations in your project.

To modify the file 'Copy\_DISDO.cmd':

- Right-Click on the Desktop Icon 'Copy\_DISDO' and select 'Edit' from the context menu.
- This cmd files uses temporary network shares to copy to the target OSs. It is prepared to serve up to eight OSs, while the copying is only active for OS1 & OS2. Modify this cmd file (i.e. add or remove 'REM' comment statements).
- If you did not stick to the OS naming scheme during Windows setup (as required by the IMAC L Installation Guideline), you will have to rename the shares in this cmd file as well.
- If you missed to enable the sharing-access to all OSs 'C' drives (see IMAC L installation procedure), this copy will fail. In this case please fix your installation.
- Save the cmd file
- Thoroughly test, if all files are copied correctly to all OSs.

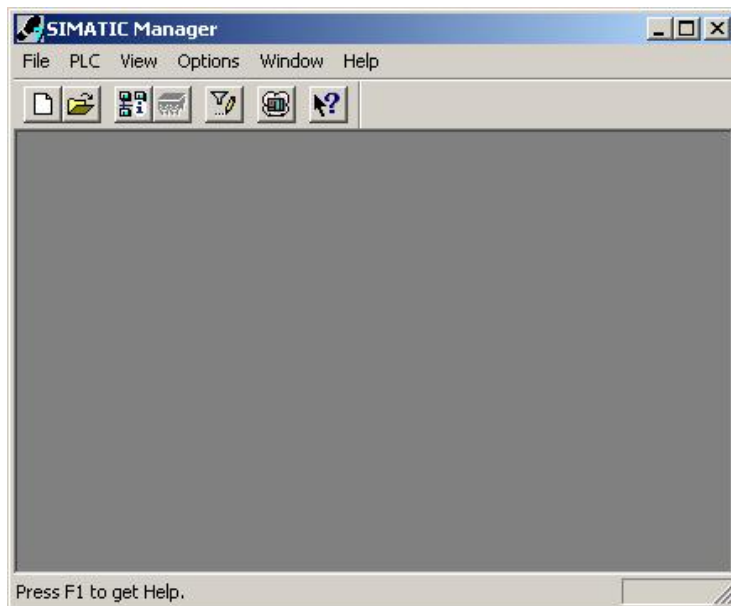
### 7.9 Import of AbvSPU Code into the Simatic S7 Manager Software

**Hint :** If you have more than one PCU in your project, this sequence has to be repeated for every PCU in your system.

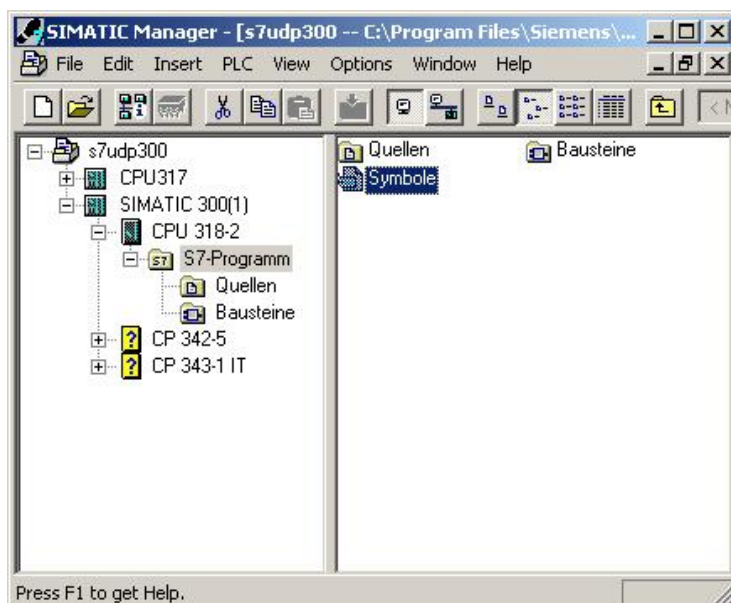
- Double-click on the 'Simatic Manager' icon on your desktop:



- The Simatic Manager window will open :



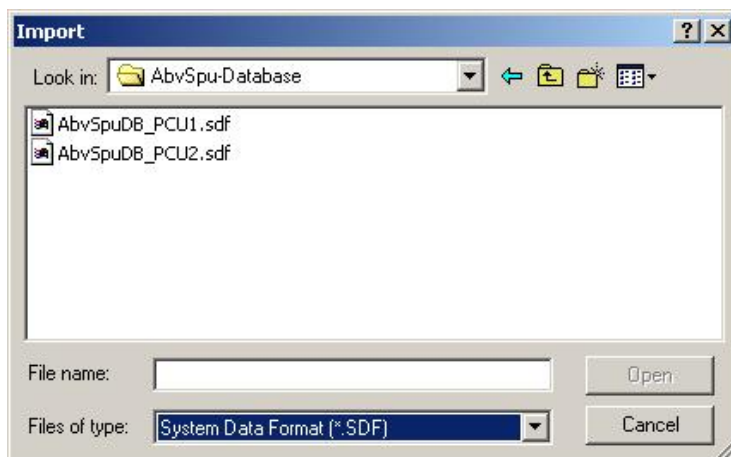
- Click File → Open and select the IMAC L Template file (as it has been unpacked during the installation sequence)
- Open the Program tree and double-click on S7-Programm → Symbole



- (The Symbol Table will open)
- Click Symbol Table → Import

Status	Symbol	Address	Data type	Comment
7	abvspu_AGxRECV	FC 146	FC 146	UDP-AGxRECV
8	abvspu_AGxSEND	FC 145	FC 145	UDP-AGxSEND
9	abvspu_AlmGrpCou...	FC 117	FC 117	Bearbeitung Alarmzaehler <Visual Output Group
10	abvspu_AMP	FC 111	FC 111	Analoge Messwertverarbeitung
11	abvspu_AMPSIM	FC 112	FC 112	Stoergroessenausschaltung
12	abvspu_BSND	FB 12	FB 12	Senden via BSEND
13	abvspu_BSND_IDB	DB 192	FB 12	Instanz-DB BSEND
14	abvspu_ChkPvObj	FC 121	FC 121	PV-Adresse pruefen
15	abvspu_COPY_DB	FC 102	FC 102	Kopiere DB-Bereiche
16	abvspu_CP_RECV	FC 130	FC 130	CP-Recv
17	abvspu_CP_SEND	FC 131	FC 131	CP-Send
18	abvspu_DB_AKKU3_4	DB 140	DB 140	DB akku3_4

- Select the symbol table source file (\*.sdf) generated before by AvET (in path C:\vision\prjAvET\AbvSpu-Database). One symbol table for every PCU will be located in this folder. Select the right file for the PCU you are currently generating :

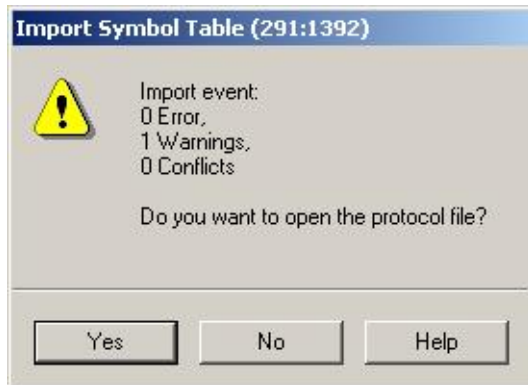


- Acknowledge, that 'undo' will not be possible by pressing 'Yes' :





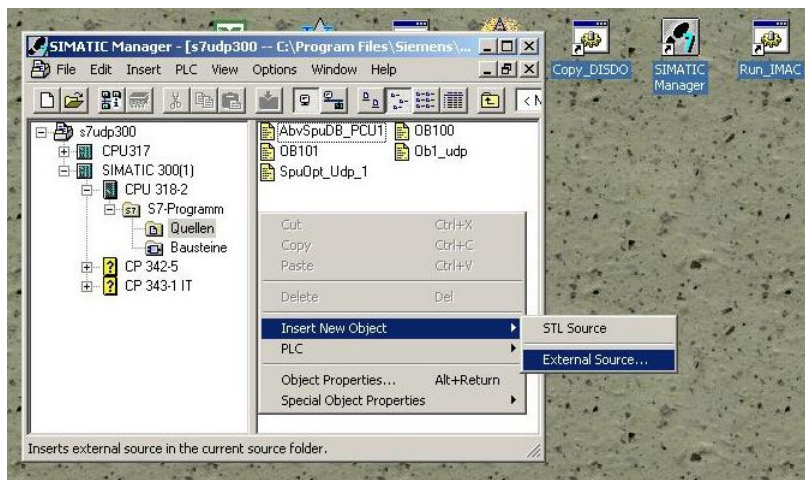
- The result window should display '0 Errors'. Warnings can usually be ignored (resulting from symbols, which were already existing from prior importing). Select 'No' :



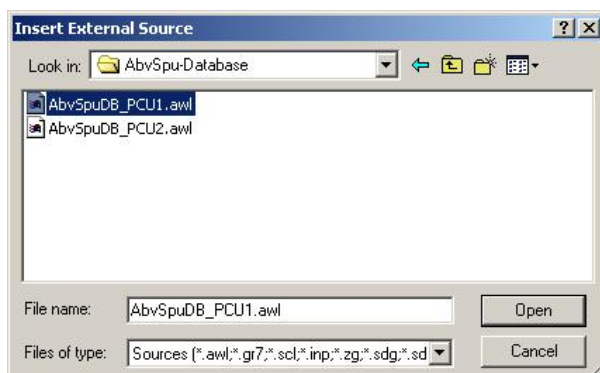
- Save changes to the symbol table by clicking on the 'Save' Icon of the Symbol Editor :



- Close the Symbol Editor.
- In the Simatic Manager Click on 'S7-Programm → Quellen'
- In the right window make a 'right-click' and select 'Insert New Object → External Source' :

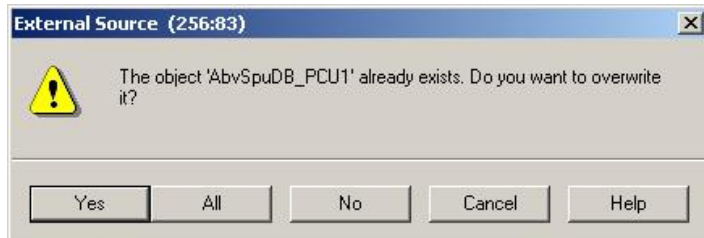


- Select the S7 source file (\*.awl) generated before by AvET (in path C:\vision\prjAvET\AbvSpu-Database). One source file for every PCU will be located in this folder. Select the right file for the PCU you are currently generating :

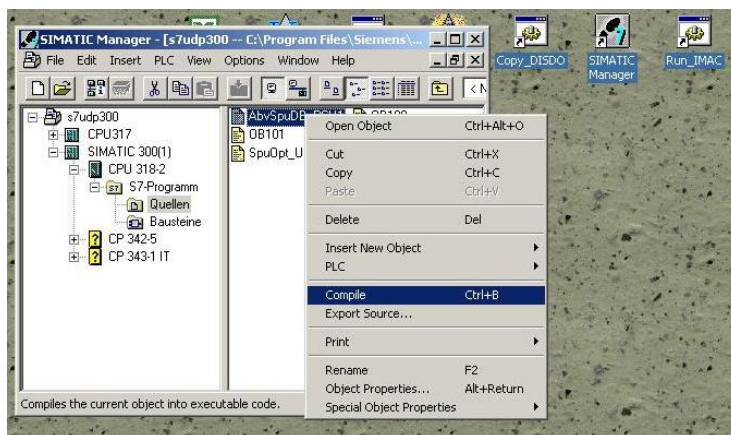




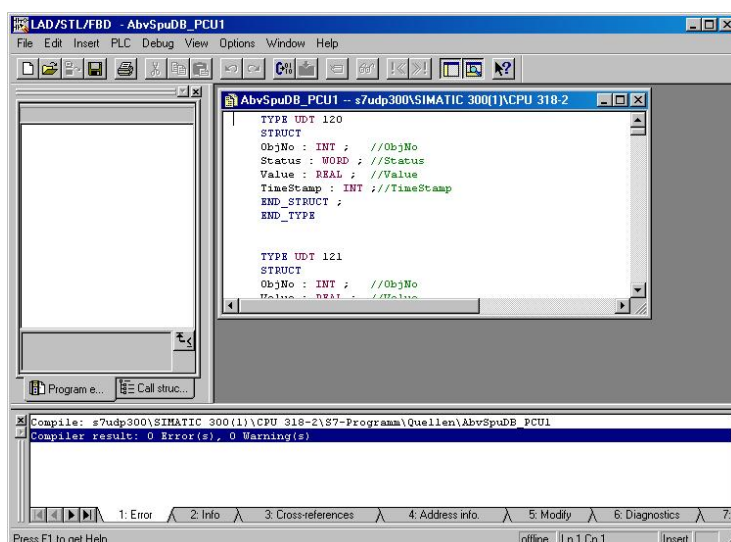
- If you have previously imported the source file for this PCU, the following window will ask, if the existing file is to be replaced :Acknowledge the replacement of the old file by pressing 'Yes' :



- In the right window right-click on the file 'AbvSpuDB\_PCUx' (x = PCU-number). Select 'Compile' :



- (The compiler window will open)
- The result will be displayed (0 Errors, 0 Warnings). Any warnings or errors have to be tracked and remedied (e.g. invalid entry in Excel field 'Address' the S7-compiler cannot compile).

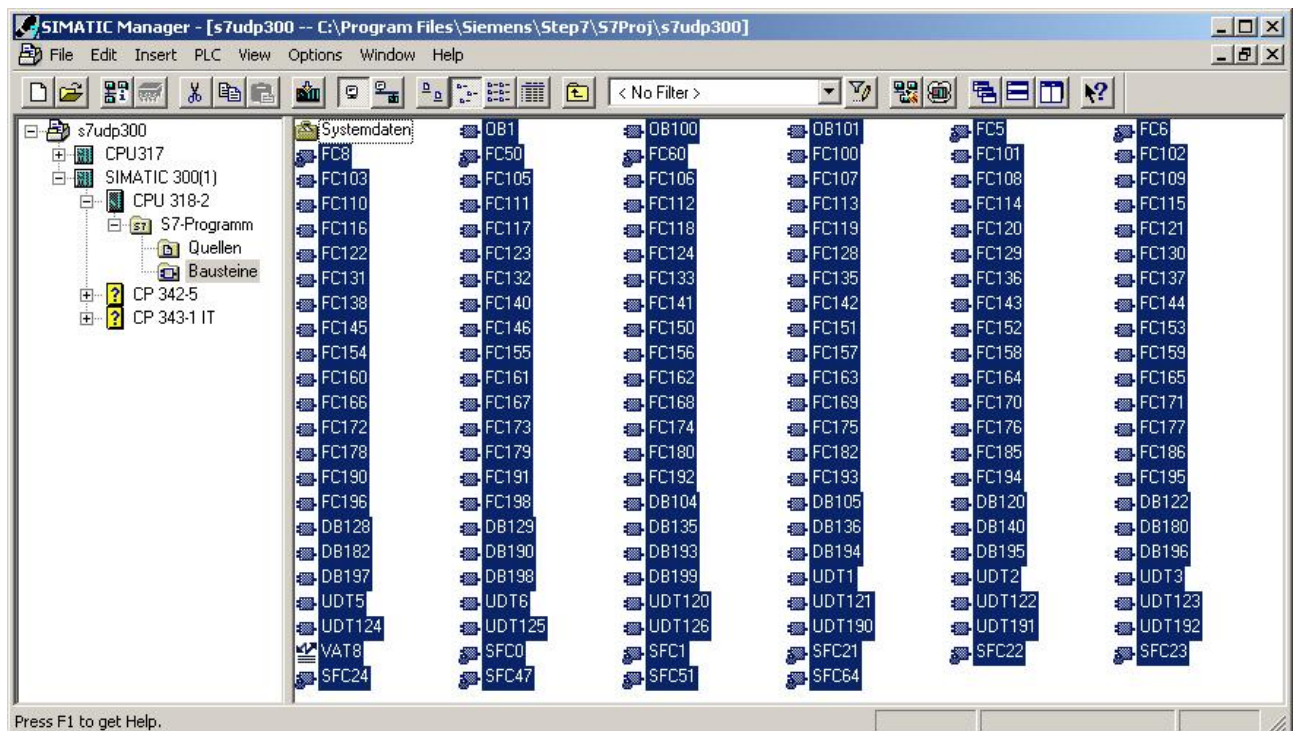



- Close the compiler window.

### 7.10 Transfer of Simatic S7 AbvSPU Code to the PCU(s)

**Hint :** If a PCU will be loaded for the first time, for ome older CPUs you will require a 'MPI' type of connection to transfer the Hardware configuration for the first time (not possible using an Ethernet connection via 'Softnet').

- Set the PCU to be loaded to 'Stop' (manual switch on CPU or using the PG-function)
- Select 'S7-Programme → Bausteine'.
- Select all objects except 'Systemdaten'. For the first download after a modification of the Simatic 'Hardware configuration' or 'NetPro configuration', please include 'Systemdaten' as well.



- Click on the Download Icon of the Simatic Manager ().
- Acknowledge to replace all existing files (from prior downloads).
- Set the PCU to be loaded to 'Run' (manual switch on CPU or using the PG-function)

## 8 Engineering in AvET

Parts of the Engineering of IMAC L have to be done in AvET. The following sections will describe the necessary procedures. All of these functions are basically just optional. The engineering as described will only be necessary, if the respective functions are required. Nevertheless, experience says, that most of these functions are required to build an average IMAC L ship's monitoring system.

### 8.1 Blocking Groups

BLOCKING GROUPS are required to block alarms depending on the status of a 'source' MePo. A typical example is the blocking of alarms of a specific diesel engine, if this engine is stopped. In this operating condition (stopped) a whole group of alarms may become 'undesired', because the engine is not in operation (e.g. Lube Oil Pressure 'Low' status). Any alarms which are now undesired, can be assigned to the 'Stopped' blocking group of this engine. These alarms will be suppressed, as long as the engine is 'stopped'. The status of MePos with a currently suppressed alarm will show 'Blocked' to indicate that usually there would have been an alarm, but it has been suppressed due to blocking.

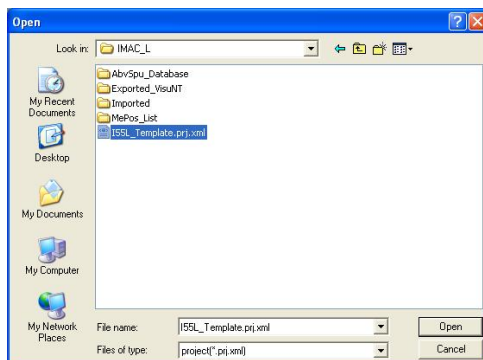
This section describes, how a BLOCKING GROUP can be defined in AvET. This will include defining, which status(es) of which MePo(s) describe the active condition of this BLOCKING GROUP. All MePos in the Excel MePo Front-end can then refer to a BLOCKING GROUP once defined in AvEdit to block one or more of their statuses [→ 5.3.32 Fields : 'BLG' on page 51]. If the respective BLOCKING GROUP is later on in its 'active' condition, all MePo statuses that have referenced this BLOCKING GROUP in their MePo data set, will not cause an alarm any more.

**Important Restriction :** For IMAC L all MePos used in conjunction with one specific BLOCKING GROUP have to be allocated in the same PCU ! (All Blocking Group processing is done within the AbvSPU Signal Processing Unit of the respective PCU).

If it is necessary to incorporate Information from other PCUs, the required data from / to other PCUs have to be transported by a communication link. The programming of this link (send-receive) is part of the user-defined technology program. IMAC L will not generate code for this cross-communication. Data received by a PCU on a link of this kind usually have to be entered as new 'internal' MePos (i.e. data source is in a DB) in the MePo list of this PCU.

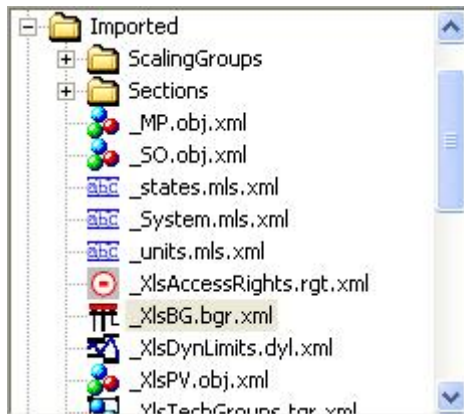
To define or modify a BLOCKING GROUP, proceed as follows :

- Double-Click on the AvET (AlphaVision Engineering Tool) Icon on your Desktop
- Click on 'File → Open'



- Point to file C:\vision\prjAvET\IMAC\_L\I55L\_Template.prj.xml
- Click on 'Open'

- Double-Click on the 'Imported' Folder
- Double-Click on the '\_XlsBG.bgr.xml' File



- In the middle window a list of all previously defined BLOCKING GROUPS will be displayed.
- To modify an existing BLOCKING GROUP, just click on the appropriate line in the middle window.
- To add a new BLOCKING GROUP, click on 'New Object' (🌐) in the Menu Bar. A new BLOCKING GROUP will appear with a default name.
- Then edit the BLOCKING GROUP parameters in the right window as required (see parameter description below).

Name	1 Test Blocking 1
Description	
Catenation PntId	1
PntId1	010101 (Test Point Binary (Ind.))
State1	1
PntId2	
State2	0
PntId3	
State3	0
PntId4	
State4	0
DstPntId	_BG01 (Blocking Group 01)

- Click on the 'Apply' button (📄) in the right window to save the changes to this list.
- Click on the 'Save' button (💾) of AvET.
- The new Blocking Group is now ready for use :

Name	Description	Catenation PntId	PntId1	State1	PntId2	State2	PntId3	State3	PntId4	State4	DstPntId
E... 📄	Enter text here	📄 Enter text ...	📄 Enter text here	📄 E...	📄 E...	📄 E...	📄 E...	📄 E...	📄 E...	📄 E...	📄 E...
1	Test Blocking 1	1	010101 (Test Point Binary (Ind.))	1		0					

The following parameters are offered for a BLOCKING GROUP:

- Name** : Identifier of this BLOCKING GROUP. In IMAC L this name has to be '01' to '99' for the range of permissible BLOCKING GROUPS. Other (alphanumeric) names can be entered here, but cannot be referred to from the IMAC L MePo Front-end.
- Description** : Text describing the purpose of this Blocking Group
- Cantenation PID**: Logical connection of the source MePos. Up to four MePos (Pntld1 – Pntld4) can be used to compute the 'active' condition of this BLOCKING GROUP. The index parameter specifies, how the logical connection between these four MePos is actually computed. The choice '1' in this example means 'use just use Pntld 1'. Other possible values for this logical connection can be found in the drop down menu on the right of this field.
- Pntldx (x=1-4)**: Point ID of the Source MePos used to compute the 'active' condition of this BLOCKING GROUP. The index parameter (see above) specifies, how the logical connection between these four MePos is actually computed. Pntldx entries not used in the computation as specified in 'Index' to be left empty.
- Statex (x=1-4)**: Status of MePo Pntldx, that causes this MePo to be computed as active (true) in the computing of the 'active' condition of this BLOCKING GROUP. If the MePo Pntldx is not in this status, this MePo will be computed as inactive (false) in the computing of the 'active' condition of this BLOCKING GROUP.
- DstPntld** : Destination Point ID. This Point is necessary to store the result of the Blocking Group. Each Blocking Group requires the reference to one internal (binary) MePo. This MePo has to be entered in the MePo Frontend specifically for this purpose and may not be used for any other purposes.

In the example as shown above, the BLOCKING GROUP '01' will be in its active condition, if the MePo named '010101' is in its status '1'. For any other status of this MePo the BLOCKING GROUP will be inactive.

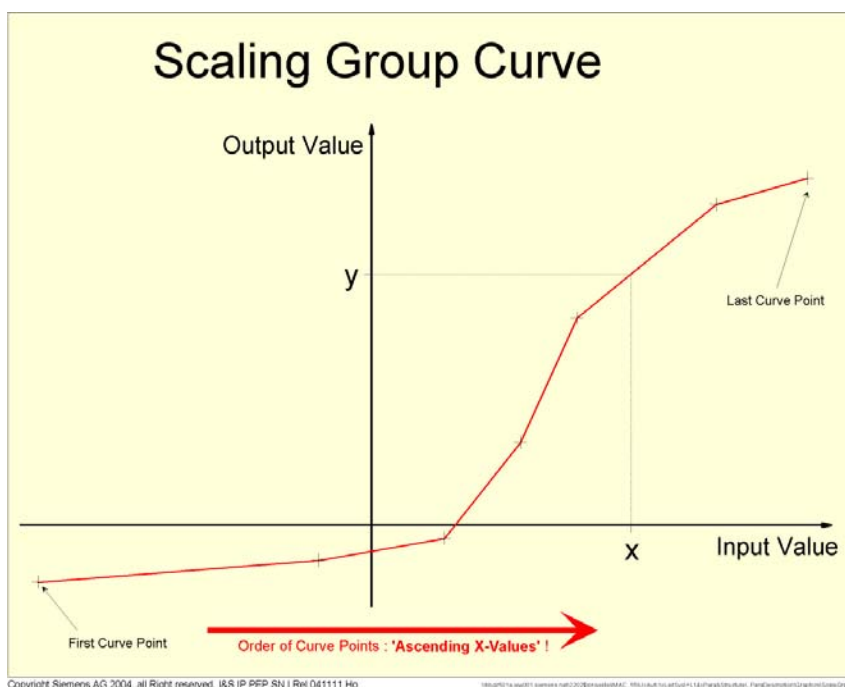
After entering this information, the BLOCKING GROUP '01' can now be referred to by any MePo in this PCU by entering '01' to the 'BLG' field in the MePo Front-end [→ 5.3.32Fields : 'BLG' on page 51].



### 8.2 Scaling Groups

A Scaling Group is basically a free definable 'curve' with multiple curve points (min. 2) . Scaling Groups are used in IMAC L for two purposes :

1. Normalization, linearization, scaling or zero point adjustment of analogue input values (which is described in this section)
2. Dynamic limit computing in DYNAMIC LIMIT GROUPS (e.g. Exhaust Gas Mean Value Deviation). The use of SCALING GROUPS for computing DYNAMIC LIMITS is explained in Section 8.3 Dynamic Limit Groups on page 98.

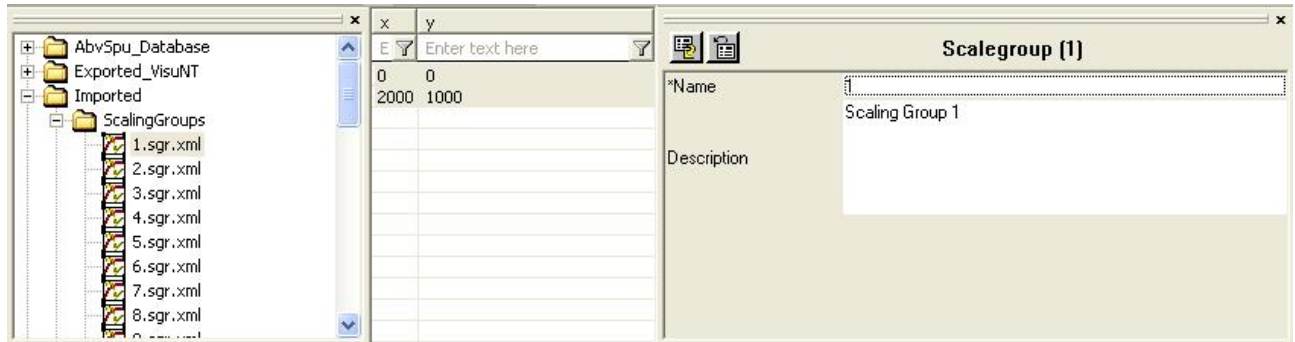


To define or modify a SCALING GROUP, open this group as follows :

- Double-Click on the AvET (AlphaVision Engineering Tool) Icon on your Desktop
- Click on 'File → Open'
- Point to file C:\vision\prjAvET\IMAC\_L\I55L\_Template.prj.xml
- Click on 'Open'



- Double-Click on the 'Imported' Folder
- Double-Click on the 'Scaling Groups' Folder
- The left Window will show a list of all Scaling Groups already existing :





### To edit an existing SCALING GROUP curve point :


- Double-Click on the 'xx.sgr.xml' file of the Scaling Group to be modified.
- The middle Window will show all curve points of this SCALING GROUP.
- Click on the curve point to be modified
- Edit parameters as desired (Description, x, y) in the right window.

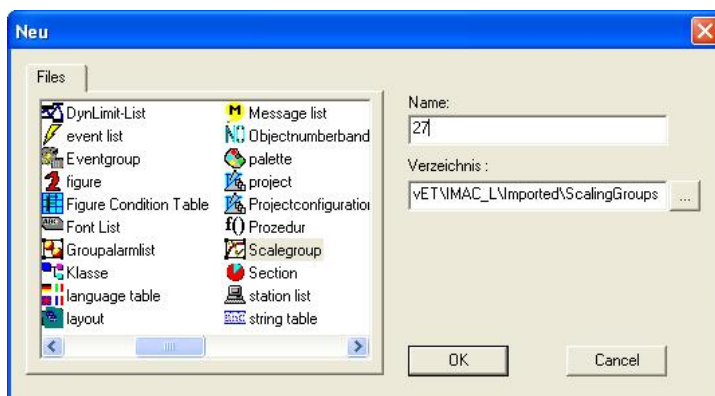
**Attention : Curve points have to be sorted in ascending order of 'x'-Values !**

**Attention : At least two curve points are required to make a valid curve !**

- Click on the 'Apply' button (  ) in the right window to save the changes to this list.
- Click on the 'Save' button (  ) of AvET.

### To insert a new SCALING GROUP :

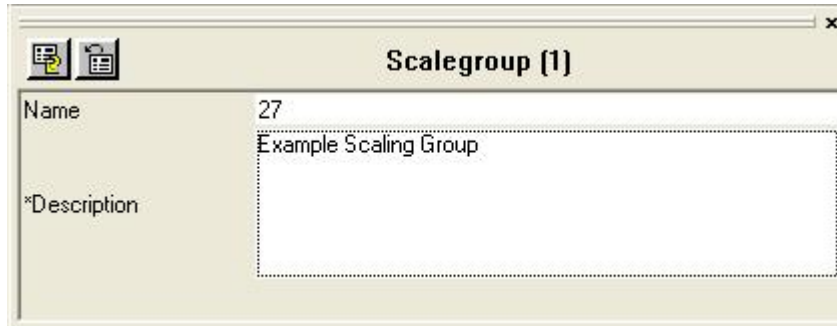
- Click on the 'Scaling Groups' Folder
- Click on 'New' (  ) in the Menu Bar.
- The 'Neu' (New) dialogue will appear:



- Select 'Scalegroup' from the list of available filetypes
- Enter the Name of the SCALING GROUP. In IMAC L this name has to be '1' to '99' for the range of permissible SCALING GROUPS. Other (alphanumeric) names can be entered here, but cannot be referred to from the IMAC L MePo Front-end.



- Leave the 'Verzeichnis' ('Directory') input field as it is (points to the 'ScalingGroups' folder).
- Click on 'OK'
- In the right Window the Name and Description of the new SCALING GROUP will be displayed:



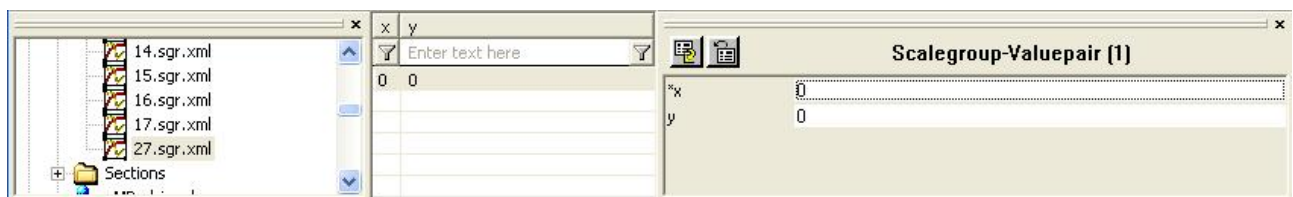
- Enter a description Text that describes the purpose of this SCALING GROUP.
- Click on the 'Apply' button (icon) in the right window to save the changes to this list.
- Click on the 'Save' button (icon) of AvET.
- A new Scaling Group exists now, but does not yet contain curve points !

### To insert a new Curve Point for an existing SCALING GROUP :

- Double-Click on the 'Scaling Group'
- Click on 'New Object' (icon) in the Menu Bar. A new Curve Point will appear in the middle Window as a new line.
- Click on the new curve point line, that appeared in the middle window.
- Edit the Curve Point parameters (Name, x-Value, y-Value) in the right window.

**Attention : Curve points have to be sorted in ascending order of 'x'-Values !**

**Attention : At least two curve points are required to make a valid curve !**



- Click on the 'Apply' button (icon) in the right window to save the changes to this list.
- Click on the 'Save' button (icon) of AvET.

**Example:** After entering a SCALING GROUP named '27' with at least two curve points, this SCALING GROUP can be referred to by any MePo by entering '27' to the 'ScaleGroup' field in the MePo Front-end [5.3.29 Field : 'ScaleGroup' on page 49]. The input Value 'x' for any MePo referring to the SCALING GROUP '27' will be taken directly from the respective MePo's data source. For an analogue hardware input channel this is directly the binary coded number as it is generated by an analogue input module! This means, that the usual transmitter type based encoding will not be performed for MePos with a ScalGrp. All transmitter-based encoding has to be included in the curve!. The output value 'y' is the result that is computed from the input by applying the curve. This value 'y' will then be used as the MePos value.

**Attention:** if a SCALING GROUP is used to directly compute a MePo's value by setting the 'Scale-Group' field in the MePo Front-end [5.3.29 Field : 'ScalGroup' on page 49], it is nevertheless necessary to refer to a TRANSMITTER TYPE in the field 'Transmitter' [→ 5.3.20 Field : 'Transmitter' on page 45]. This is because the scaling curve will only be used to compute the numerical result for the indication of the MePo's value, while the TRANSMITTER TYPE entry will at the same time be used to perform the 'Out Of Range' detection for such points.

The processing for such points directly using a SCALING GROUP is:

- The input signal (e.g. from an analogue input module) will be processed by the SCALING GROUP curve to calculate the value of the MePo.
- The same input signal will be processed by the TRANSMITTER TYPE curve to detect, if 'Out of Range' is active for this MePo.
- Bar graph indications in IMAC L (e.g. in the POINT REPORT) resulting from a calculation in a SCALING GROUP curve will be indicated using the Range limits as set in the 'Range Start' [5.3.23 Field : 'Range Start' on page 46] and 'Range End' [5.3.24 Field : 'Range End' on page 46] fields. This means that for a correct analogue bar graph indication it is in this case necessary, that the computed numerical results from the SCALING GROUP will fit into the range as set by these two fields.

As a result of this, it may be necessary for special / exotic curve shapes in a SCALING GROUP (e.g. not just a minor linearization but a completely new scaling / recalculation of a signal) to set up a specifically created new TRANSMITTER TYPE in order to correctly set the 'Out of Range' limits for such a computed MePo [see 8.4 Transmitter Types on page 102].

### 8.2.1 EXCEL 'Tank Curve' Tool

For ships one very common use of Scaling Groups is for 'Tank Form Curves'.

Usually the measurement of tank contents on a ship uses a tank level sensor (i.e. filling height) as the primary sensor. A very common measurement for tank levels is using an analogue differential pressure transducer. The detected pressure is proportional to the filling height (level).

To operate a ship it is necessary to have a volume indication of the tank contents – and not it's filling level. As an example, only the volume of fuel contained in a tank will determine, how long this quantity of fuel will last for ship's operation. An indication of the filling level would be useless (except for bunkering - to determine the 'full' condition of a tank).

So it is necessary to convert the measured 'level' to 'volume'. As most ship tanks have quite irregular shapes, there is in most cases no linear correlation between a tank's filling level and the media volume currently being contained in this tank.

For this purpose, a yard usually supplies a so called 'Tank Metering Table' for each individual tank. This table provides us with the relation between 'filling level' and 'contained volume'.

Basically it would be possible to enter the data from all 'Tank Metering Tables' directly to AvEdit Scaling Group' curves (one curve for each tank). But this has some serious disadvantages:

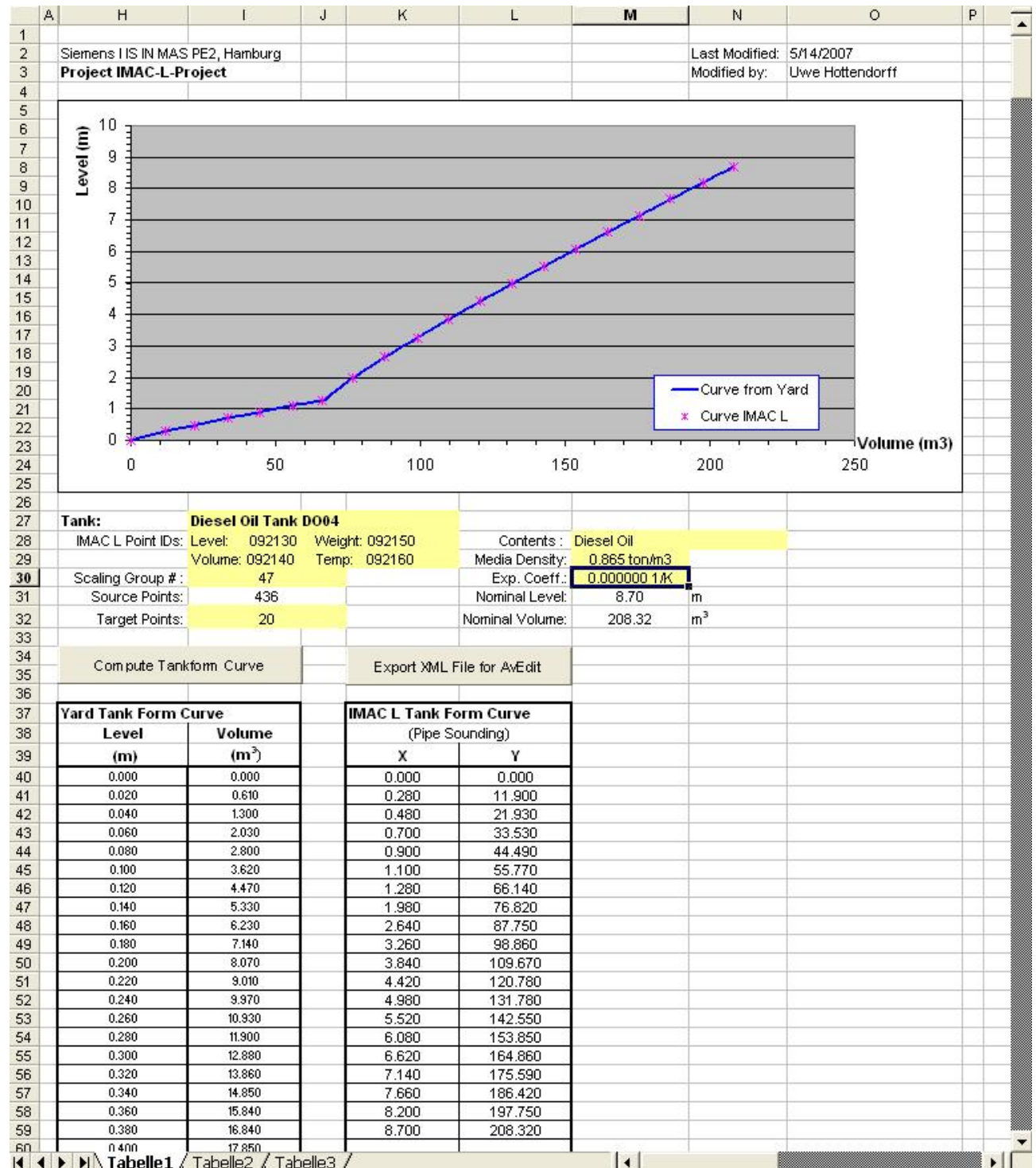
- A yard usually provides hundreds of curve points per tank. IMAC L can't handle that many curve points. The number of points would have to be reduced manually (interpolation using a pocket calculator) to achieve a reasonable number of curve points (up to ~ 20 per tank).
- There is very much typing and calculation work to be done for this solution. This typing is expensive and prone to error.
- Errors in curves are difficult to detect in commissioning (tank contents has to be simulated).

For IMAC L there is a simple solution to avoid these disadvantages and have an easy and convenient engineering for tank curves. The IMAC L **"Tank Curve Tool"** (EXCEL spreadsheet, one per tank) is used to convert the yard's 'Tank Metering Tables' to IMAC L Scaling Group curves.

Your IMAC L CD contains this spreadsheet as a template:

**'IMAC\_TankCurve\_Tool\IMAC\_L\_Tankform\_Curve.xls'**

This Spreadsheet looks like this:



To use this Excel sheet, please proceed as follows:

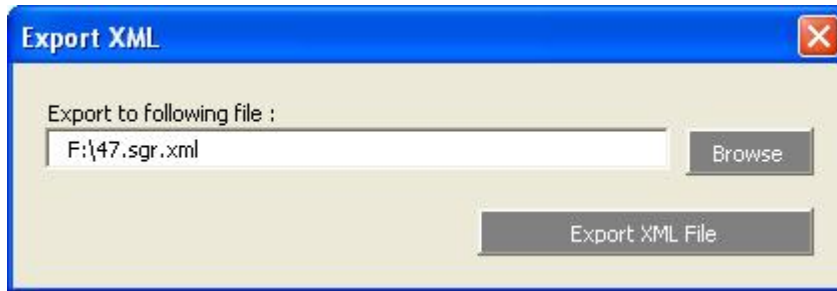
- Create one copy of this sheet for every tank that needs a Tank Form Curve. Make the Naming of the files contain the Scale Group Number and the tank's name.
- Put the yard's 'Tank Metering Table' data (level / volume pairs) to the table at the left bottom (e.g. copy/paste if possible). Start with cells "H40:I40" for the first pair. Fill in all pairs while leaving no gaps (empty cells). Have a numerical value in each used cell.
- Fill in the Header of the Table (yellow fields). The fields in the header are:
  - **"Tank"**: The name of the Tank (ends up as 'Comment' in AvET Scaling Group)
  - **"IMAC L Point IDs"**: Point ID names in IMAC L, which are used in conjunction with this tank. No direct use, but nice for documentation (printing this EXCEL sheet will deliver all information on the handling of this tank in IMAC L).
  - **"Scaling Group #"**: The number of the AvET Scaling Group number that is used for this tank. This number has to be unused (or reserved for this tank) in AvET and has to be unique in your project. Leave no gaps in Scaling Group numbering.
  - **"Target Points"**: The number of Points to be used in the IMAC L curve for this Tank. The yard's 'Tank Metering Table' data will be converted to a reduced set of points using linear interpolation. This is the number of points used for this IMAC L Scaling Group curve. Start with a default value (e.g. '5') for "Target Points".
  - **"Contents"**: Type of media stored in this tank. No direct use, but nice for documentation (printing the EXCEL sheet will deliver all information on the handling of this tank in IMAC L).
  - **"Media Density"**: Density of media stored in this tank. No direct use, but nice for documentation (printing the EXCEL sheet will deliver all information on the handling of this tank in IMAC L).
  - **"Expansion Coefficient"**: Expansion factor for media stored in this tank (volume increase per 1 Kelvin temperature rise). No direct use, but nice for documentation (printing the EXCEL sheet will deliver all information on the handling of this tank in IMAC L).
- Now click on the **"Compute Tankform Curve"** button. The IMAC L tankform curve will be immediately computed from the yard curve. The target curve will have as many points as specified in "Target Points" field. The result is displayed as a table (level / volume pairs at the right bottom; starting with cells "K40:L40" as the first pair) and as a diagram at the top.
- Now the number of curve points (field "Target Points") has to be optimized. Compare both curves in the diagram. Try to find a good match between both curves while in the same time trying to have as small a number of "Target Points" as possible. For every modification of "Target Points" simply click again on the button **"Compute Tankform Curve"**.
- Click on the "Export XML File for AvEdit" button.

This export will create an XML file, which can be directly copied to the following AvET directory:

**“C:\vicon\PrjAvET\Imported\ScalingGroups\”**

AvET has to be closed, before you copy generated files into this directory !

A dialogue will prompt you for the destination directory & file name:



You may adjust the target drive & directory to fit your purposes. The file name will be automatically derived from the “Scale Group number” field. Do not change this default name! (the file name specified has to fulfill the IMAC L naming requirement for IMAC L Scaling Groups)

**“xx.sgr.xml” (where xx is the “Scaling Group number”)**

The example above shows the export of the Scaling Group with number ‘47’.

- Click on **“Export XML File”** to start creating the required file.

Successful creation of the file will be reported:



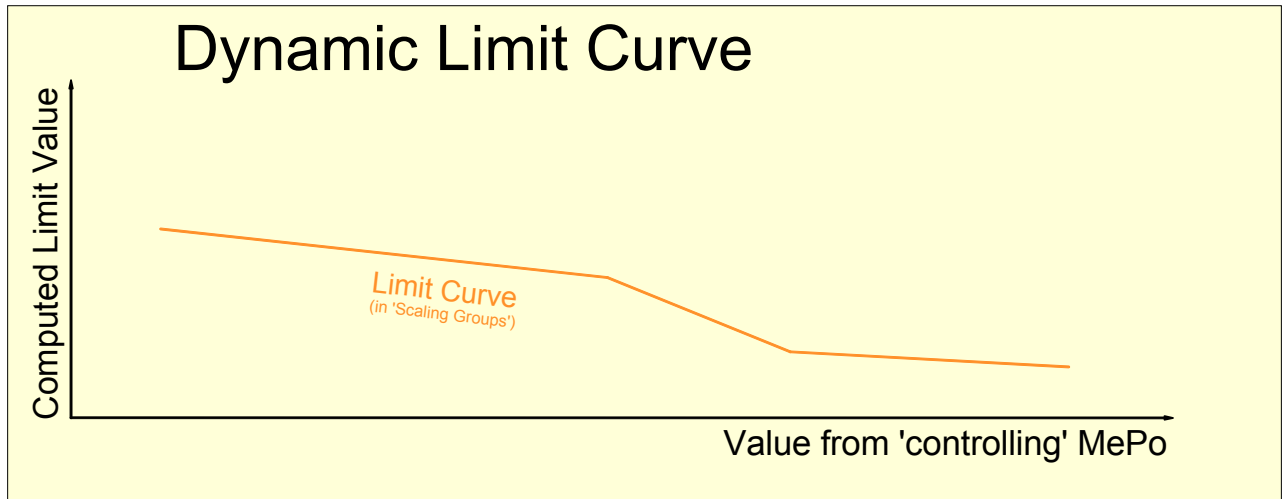
The created file is ready for use in AvET.

- Print your Spreadsheet (PDF, Paper) for your project documentation
- Save your Spreadsheet & close EXCEL
- Close AvET (if it should be opened)
- Copy the XML file(s) manually to the a.m. directory (if you didn't export it (them) directly to the required destination).
- Open AvET and check the resulting Scaling Group curve.
- Use the **“Scaling Group numbers”** for tanks in the IMAC L MePo List (enter in field '5.3.29 Field : 'ScalGroup') to do the required tank volume computing.
- Use one additional MePo (without ScaleGroup) to indicate the filling level of the tank as well.



### 8.3 Dynamic Limit Groups

Dynamic Limit Groups are used to compute a dynamic limit value depending on the value of an analogue MePo. A typical example are exhaust gas deviation limits. These limits are more tolerant for low exhaust gas mean values and have to be more restrictive for high exhaust gas mean values.



To achieve this dynamic function for limits, IMAC L will use SCALING GROUPS [→ 8.2 Scaling Groups on page 91]. The input ('x') -value of the curve will be the value taken from a controlling MePo. In the example of the exhaust gas deviation limits, the controlling MePo will be the 'Exhaust Gas Temp. Mean Value' MePo. The result ('y') -value of the curve will be stored in the DYNAMIC LIMIT GROUP itself.

Any MePo referring to this specific DYNAMIC LIMIT GROUP by entering the number of the DYNAMIC LIMIT GROUP to its 'Dyn Limit Group' field [→ 5.3.39 Fields : 'Dyn Limit Group' on page 56] will use the limit value computed in the DYNAMIC LIMIT GROUP, and not the value from the 'Limit(s)' field [→ 5.3.38 Fields : 'Limit(s)' on page 55]. Nevertheless the 'Limit(s)' field must be filled in for initial operation (until a valid dynamic limit has been computed once).

There are two basic steps required in order to make a DYNAMIC LIMIT GROUP work :

1. Create the required Limit Curve(s) as a separate SCALING GROUP. This step is already explained in detail above in this document [→ 8.2 Scaling Groups on page 91].
2. Create a DYNAMIC LIMIT GROUP in AvET, which makes the connection between the 'controlling' MePo and the curve from the Scaling Groups to be used to calculate the limit.

**Important Restriction :** For IMAC L all MePos used in conjunction with one specific DYNAMIC LIMIT GROUP have to be allocated in the same PCU (Source MePo and all MePos using this dynamic limit) ! All DYNAMIC LIMIT GROUP processing is done within the AbvSPU Signal Processing Unit of the respective PCU. The Scaling Groups defined in conjunction with Dynamic Limit Groups may be defined once and can be used for multiple Dynamic Limit Groups in any PCU !

If it is necessary to incorporate Information from other PCUs (i.e. the Source MePo), the required data from other PCUs have to be transported by a communication link. The programming of this link (send-receive) is part of the user-defined technology program. IMAC L will not generate code for this cross-communication. Data received by a PCU on a link of this kind usually have to be entered as new 'internal' MePos (i.e. data source is in a DB) in the MePo list of the receiving PCU.



**Important Restriction regarding the numerical order of Limits:**

All limit values in IMAC L have to be assorted in descending order of numerical values. See Section 5.3.38 Fields : 'Limit(s)' on page 55 for more information on this basic restriction in IMAC L. This restriction is true for fixed limit values AND for dynamic limit values being calculated using the DYNAMIC LIMIT CURVES.

If a DYNAMIC LIMIT is set in a way, that the resulting limit value would violate this rule, the DYNAMIC LIMIT will not be set. In this case the limit will stay at the last valid value that has been computed before. If there was no valid computation before, the default value from the 'Limit(s)' field will stay active.

This condition might then look as if the DYNAMIC LIMIT function does not work. Please carefully check your dynamic limit settings with respect to this restriction and carefully look for possible conflicts before reporting malfunctions of the DYNAMIC LIMIT function !

It is strongly recommended to use one of the following three strategies to avoid this type of limit value conflicts:

- Use only one DYNAMIC LIMIT and no fixed limit for one MePo. With only one limit value, there is no possible conflict in the descending order of limit values.
- If more than one DYNAMIC LIMIT is required for the same MePo, make sure that all of the DYNAMIC LIMITS will depend on the same 'controlling MePo' and that the curve shapes of all DYNAMIC LIMIT CURVES used are in the required descending order of values (for each possible 'controlling' input value). This means, that the curve shapes used for the limits do not intersect or touch ! Do not use fixed limits in combination with the DYNAMIC LIMITS.
- If the intersection of limit curves cannot be avoided or if the controlling MePos of two or more curves are completely independent from each other, more than one alarming MePo has to be created. Each of these MePos shall use only one of the conflicting curves and no fixed limits. This ends up in the limits being totally independent from each other and thus no conflicts can occur. The drawback on this strategy is that the operator will get alarms from more than one MePo, even if it is from the technological point of view the same object / situation, that is causing the alarms.

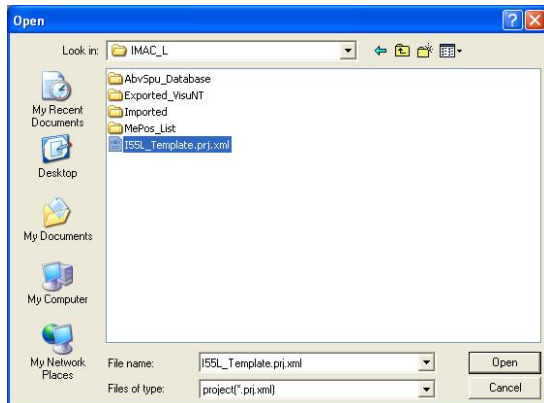
The following method (although looking OK) is potentially dangerous and thus not recommended:

- Make sure (by carefully checking the DYNAMIC LIMIT CURVE AND the range of the controlling MePo), that there are no possible conflicts with the neighboring fixed / dynamic limit values above AND below of the DYNAMIC LIMIT being calculated.

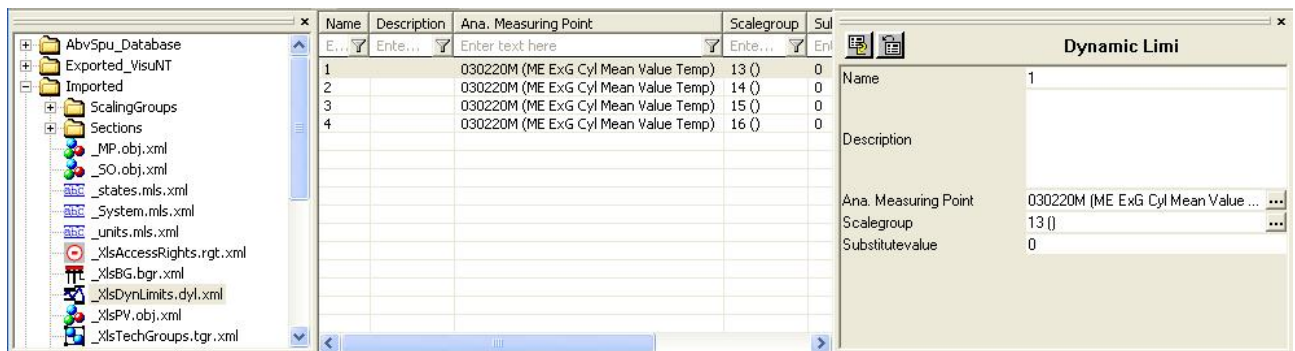
This solution is dangerous, because fixed limits can be modified by operator input. If the operator input is not consistent with the a.m. restrictions, this will cause a conflict during runtime. This can cause a malfunction of the DYNAMIC LIMITS Function. Limits not working as designed have to be regarded a potentially dangerous situation !

To define or modify a DYNAMIC LIMIT GROUP, open this group as follows :



- Double-Click on the AvET (AlphaVision Engineering Tool) Icon on your Desktop
- Click on 'Datei → Öffnen' (File Open)
- Point to file C:\vision\prjAvET\IMAC\_L\I55L\_Template.prj.xml
- Click on 'Open'






- Double-Click on the 'Imported' Folder
- Click on the '\_XlsDynLimits.dyl.xml' file
- The middle Window will show a list of all DYNAMIC LIMIT GROUPS already existing :

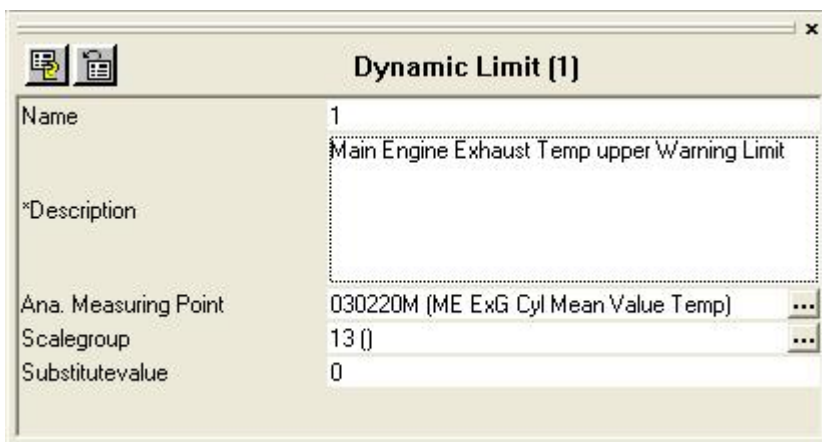


To edit an existing DYNAMIC LIMIT GROUP :

- Click on the '\_XlsDynLimits.dyl.xml' file
- The middle window will show all DYNAMIC LIMIT GROUPS defined in this project.
- Click on the DYNAMIC LIMIT GROUP to be modified in the middle window
- Edit parameters as desired in the right window.
- Click on the 'Apply' button (  ) in the right window to save the changes to this list.
- Click on the 'Save' button (  ) of AvET.

### To insert a new DYNAMIC LIMIT GROUP :

- Click on the '\_XlsDynLimits.dyl.xml' file
- The middle window will show all DYNAMIC LIMIT GROUPS defined in this project.
- Click on 'New Object' (  ) in the Menu Bar. A new DYNAMIC LIMIT GROUP will appear in the middle Window as a new line.
- Click on the new DYNAMIC LIMIT GROUP line, which appeared in the middle window.
- Edit the DYNAMIC LIMIT GROUP parameters in the right window. The parameters are explained in detail below in this section.
- Click on the 'Apply' button (  ) in the right window to save the changes to this list.
- Click on the 'Save' button (  ) of AvET.



<b>Dynamic Limit [1]</b>	
Name	1
*Description	Main Engine Exhaust Temp upper Warning Limit
Ana. Measuring Point	030220M (ME ExG Cyl Mean Value Temp) ...
Scalegroup	13 () ...
Substitutevalue	0

### The parameters of a DYNAMIC LIMIT GROUP are :

#### **Name:**

Name of the DYNAMIC LIMIT GROUP. In IMAC L this name has to be '1' to '99' for the range of permissible DYNAMIC LIMIT GROUPS. Other (alphanumeric) names can be entered here, but cannot be referred to from the IMAC L MePo Front-end.

#### **Description:**

A text that describes the purpose of this DYNAMIC LIMIT GROUP.

#### **Ana. Measuring Point:**

Point-ID of the Source MePo, on which this dynamic limit is meant to depend on. The MePo specified here has to be a MP of 'analogue' type, that is defined in the Excel MePo front-end as a standard analogue MePo. This MePo has to be allocated in the same PCU as those MePos which want to used this dynamic limit.

#### **Scalegroup:**

Name of the SCALING GROUP, that contains the dependency between the source MePo (Ana. Measuring Point) and the limit value to be used. This SCALING GROUP has to be defined as described above [→8.2 Scaling Groups on page 91].

#### **Substitutevalue:**

This value is currently not evaluated by the AbvSpu. Leave the default value ('0') in this parameter unmodified.

## 8.4 Transmitter Types

IMAC L uses so called TRANSMITTER TYPES to specify the type of an input signal (see MePo front end field 'Transmitter' [→ 5.3.20 Field : 'Transmitter' on page 45]. These are used for all sensors to specify their type of input.

Technically important are these TRANSMITTER TYPES only for analogue inputs. With analogue inputs the transmitter definitions are used to assign the input range (e.g. counts from an analogue input module) to the nominal range in IMAC L. Additionally the margin (outside of the range), that will be allowed before the MePo reports 'Out of Range' will be defined in the transmitter definition.

The following description gives an overview of how the existing TRANSMITTER TYPES list can be extended by additional transmitters or how existing TRANSMITTER TYPES can be modified.

The basic definition of all TRANSMITTER TYPES to be used in IMAC L is in the MePo Front-end in sheet 'Config' in columns 'O' & 'P' :



Name	Description	Hardwaremodul typ	Lower Li
Enter text here	Enter text here	Enter text here	Enter...
AI_4_20mA	AI 4..20 mA	Analog In	0
AI_0_20mA	AI 0..20 mA	Analog In	0
AI_PT100_0_150_	Pt100 150 °C	Analog In	0
AO_0_10V	AO 0..10 V	Analog Out	0
AO_0_20mA	AO 0..20 mA	Analog Out	0
AO_4_20mA	AO 4..20 mA	Analog Out	0
AI_PT100_0_1000F	Pt100 1000°F	Analog In	-178
AINintern_OrealC	AI internal (real)	Real	0
AINPT100_O0_100C	AI Pt100 0..100 °C	Analog In	0
AINintern_OintegerC	AI internal (coded)	Int	0
DINintern	DI internal	Int	0
DIN24V	DI	Int	0
AINPM0_10V	AI +-0..10V	Analog In	-27648
AIN0_10V	AI 0..10V	Analog In	0
AINPT100_OM50_P...	AI Pt100 (-50.....	Analog In	-500
DON24V	DO	Int	0
AINNICRNINK	AI NiCrNi K-Type	Analog In	0

It is recommend to only add TRANSMITTER TYPES and to leave the predefined transmitters untouched. Column 'O' holds a transmitter name, which has to be unique for identification of the transmitter type. Column 'P' holds a comment on the type / use of the transmitter. Additional transmitters can be entered in the first empty row below the predefined ones. As soon as new types have been entered here, the transmitters are available in the 'Objekte' sheet of the MePo front end for MePo editing.

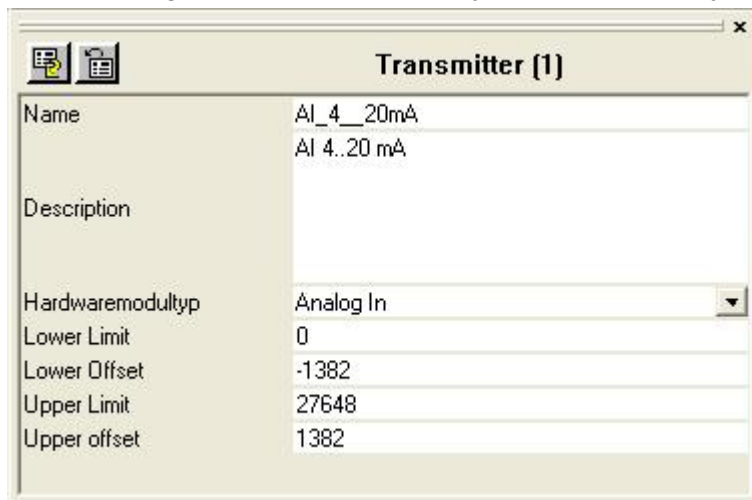
New Transmitters for analogue MePos will need additional engineering in AvEdit. To edit the properties of a new transmitter for analogue values, the 'Excel Import' function [→ 7.2 Importing MePo data from Excel on page 72] has to be performed once to read the new transmitter types(s) to the AvEdit database. Only the transmitter 'Name' and 'Description' will be imported from Excel. The other properties will be set to '0' for newly imported TRANSMITTER TYPES, while the properties of already existing TRANSMITTER TYPES in AvEdit will not be modified by importing again.

**Hint:** If Transmitters are deleted or renamed in the MePo front end, the import function will not delete the existing transmitters in the AvEdit database. In this case transmitters not required any more have to be manually deleted in AvEdit.

### To edit the properties of a transmitter Type in AvEdit :

- Double-Click on the '\_XIsTransmitters.trm.xml' file
- The middle window will show all TRANSMITTER TYPES defined in this project.
- Click on the TRANSMITTER TYPE to be modified in the middle window
- Edit parameters as desired in the right window.
- Click on the 'Apply' button () in the right window to save the changes to this list.
- Click on the 'Save' button () of AvET.

The following screen shot shows a typical transmitter type for a 4 ... 20 mA analogue signal :



### The parameters of a TRANSMITTER TYPE are :

#### **Name:**

Name of the TRANSMITTER TYPE. In IMAC L this name is imported from the MePo front end (sheet 'Config' / column 'O'). Other (alphanumeric) names can be entered here, but cannot be referred to from the IMAC L MePo Front-end.

#### **Description:**

A text that describes the purpose of this TRANSMITTER TYPE. In IMAC L this description is imported from the MePo front end (sheet 'Config' / column 'P').

#### **Hardwaremodul typ:**

may be one of the following five choices (no other values possible):

- **Analog Out**
- **Analog In (smoothed)** (with smoothing, linear scaling acc. to upper / lower limits)
- **Analog In** (no smoothing, linear scaling acc. to upper / lower limits)
- **Int** (16 bit raw value; 'integer' input value, no scaling)
- **Real** (32 bit raw value; 'real' input value, no scaling)

#### **Lower Limit:**

Input value (e.g. raw counts from an analogue module), that is assigned to the 'Range Start' of a MePo with this TRANSMITTER TYPE. If this input value is being read as an input, the 'Range Start' value in IMAC L will be indicated for this MePo.

**Lower Offset:**

Offset value (value is usually negative), that gives the margin (starting from the 'Lower Limit') by which the MePo may go below its 'Lower Limit' before an 'Out of Range' alarm will be generated by IMAC L.

**Upper Limit:**

Input value (e.g. raw counts from an analogue module), that is assigned to the 'Range End' of a MePo with this type. If this input value is being read as an input, the 'Range End' value in IMAC L will be indicated for this MePo.

**Upper Offset:**

Offset value (value is usually positive), that gives the margin (starting from the 'Upper Limit') by which the MePo may go above its 'Upper Limit' before an 'Out of Range' alarm will be generated by IMAC L.

The transmitter settings depicted above in this section are for a standard Simatic S7 analogue input module:

- This is a transmitter that delivers a 16 bit input value (no smoothing, with scaling in IMAC)
- For this transmitter 0 counts (equals to 4 mA) are 'Lower Limit / Start of Range'
- For this transmitter 27648 counts (equals to 20 mA) are 'Upper Limit / End of Range'.
- The 'Lower Offset' setting of -1382 makes the MePo go to 'Out of Range' if the input counts are below -1382 Counts (equals to approx. 3.2 mA).
- The 'Upper Offset' setting of +1382 makes the MePo go to 'Out of Range' if the input counts are above 29030 Counts (27648 Counts + 1382 counts, equals to approx. 20.8 mA).

**Example for the use of this 4...20 mA TRANSMITTER TYPE:**

If a pressure MePo for 0 ... 6 bar (Range Start, Range End, Unit) in IMAC L uses this transmitter type, the following indications will result:

- At 4 mA input (0 Counts) the point will show '0 bar'
- At 20 mA input (27648 Counts) the point will show '6 bar'
- Indications for any other input value will be computed from a linear interpolation / extrapolation out of the above mentioned two curve points.
- For input values below approximately 3.2 mA (equals to -0.3 bar or -1382 counts) the point will create an 'Out of Range' alarm.
- For input values above 20.8 bar (equals to +6.3 bar or 29030 counts) the point will create an 'Out of Range' alarm.

**Important Hint :** Simatic S7 analogue input modules will use different types of encoding for analogue signals. Please refer to the respective hardware manuals for detailed information on the encoding methods being used. Typically modules for current or voltage will have a coding scheme, which transfers the nominal range of the input signal to a numerical range of counts (e.g. 0 ... 27648 counts), while modules for measurement of temperatures will transfer their input temperature to an output, that represents 0.1 °C for each output count. It is important to understand, that the first of these methods includes a method of normalization (projection of a physical signal to a logical nominal range), while the second method is a plain temperature indication without normalization.



As analog values in IMAC L always need a nominal range for their measurement and indication, the required normalization would be missing for temperature inputs. That is compensated in IMAC L by creating different TRANSMITTER TYPES for different temperature measuring ranges. Additional TRANSMITTER TYPES for other temperature ranges or other temperature units (e.g. Fahrenheit) can easily be created as required. The following two examples show the setup scheme used with two TRANSMITTER TYPES already existing in the IMAC L templates:

**Example 1:**

A PT 100 temperature input module is being used to measure two different temperatures. One channel of this module is used to measure a room temperature (range of 0 ...50 °C) and another channel of the same module is being used for a refrigeration room temperature (range of -50 ... +50 °C). To achieve this, two different TRANSMITTER TYPES are used for the same input module. Each of these TRANSMITTER TYPES contains settings to perform the correct scaling between input (0.1°C per count) and the required output range:

The required TRANSMITTER TYPE settings for the range of 0 ...50 °C are:

- 'Lower Limit' = 0 (Range Start = 0°C → Input counts from module : 0)
- 'Upper Limit' = 500 (Range End = 50°C → Input counts from module : 500)

The required TRANSMITTER TYPE settings for the range of -50 ...+50 °C are:

- 'Lower Limit' = -500 (Range Start = -50°C → Input counts from module : -500)
- 'Upper Limit' = 500 (Range End = 50°C → Input counts from module : 500)

**Example 2:**

A NiCrNi temperature input module is used to measure a temperature with a range of 0 ...1000 °F (Fahrenheit). A standard Simatic S7 analogue module has no setting to switch its output counts to Fahrenheit (°F), so the output counts of the module are still in '0.1 °C per count'.

The required TRANSMITTER TYPE settings for the range of 0 ...1000 °F are:

- 'Lower Limit' = -178 (Range Start = 0°F = -17.8°C → Input counts from module : -178)
- 'Upper Limit' = 5378 (Range End = 1000°F = 537.8°C → Input counts from module : 5378)

**Important Hint :** If you add and use a new Transmitter Type for Analogue MePos in the EXCEL Mepo FrontEnd without adjusting the settings of this Transmitter Type in the AvET list '\_XlsTransmitters.trm.xml', this will end up in a failure while compiling the AbvSPU (division by Zero while generating the AbvSPU code).

## 9 Programming Simatic S7-Code for IMAC L Functions

For programming control functions, the standard entry point is defined in OB 1. Object numbers for PB, FC, FB and DB may be used in the range from 1 ... 79 (except for some objects already existing in the IMAC L template project (like FC5, FC6 for send/receive).

### 9.1 Undefined Groups

IMAC L will use UNDEFINED GROUPS to indicate faults for groups of Measuring Points. One unique UNDEFINED GROUP number is used to identify an IMAC L UNDEFINED GROUP. This UNDEFINED GROUP number has to be assigned to each MePo in the MePo list that is meant to belong to this group. [→ 5.3.12 Field : 'UndefGrp' on page 41]. Typical examples for the use for UNDEFINED GROUPS are serial line interfaces or peripheral I/O signals being connected via fieldbus systems like Profibus DP.

All MePos belonging to an UNDEFINED GROUP will automatically show the status 'Undefined' instead of their current process status, if the group is being set to 'undefined'. All MePos being currently undefined will be automatically visible in the HMI's UNDEFINED SUMMARY list ('XUS'). MePos currently being undefined will not cause an alarm any more. The alarming of the cause of an 'undefined' situation has to be done separately by MePos indicating the cause of the 'undefined' condition (e.g. separate binary MePo for 'Serial Interface x' with statuses 'OK' and 'FAULT').

The detection of the 'undefined / OK' status of each UNDEFINED GROUP and the function call to set this status are part of the technology program (code is not automatically generated). A simple call to FC 94 ("IMAC\_UNDEF\_SET") is used to set the 'undefined' status of an UNDEFINED GROUP:

```
CALL "IMAC_UNDEF_SET"  
UNDEF_GRP    :=17  
UNDEF_STATUS:=_serial_line_fault
```

The parameters of FC94 "IMAC\_UNDEF\_SET" are :

**UNDEF\_GRP:**

Number of the Undefined Group, for which the status is to be set. 'UNDEF\_GRP' is of type integer. The range of valid inputs for 'UNDEF\_GRP' is 1...255 in IMAC L.

**UNDEF\_STATUS:**

The Boolean input parameter 'UNDEF\_STATUS' specifies, if the Undefined Group is currently undefined [TRUE] or OK [FALSE].

In the example above the call to FC 95 will set the UNDEFINED GROUP No. 17 to the value of the Boolean variable "\_serial\_line\_fault".

Repeated (cyclical) calls to FC 95 for each undefined group are recommended to achieve a 'real-time-like' response for malfunctions of a group.

**Important Restriction :** For IMAC L all MePos used in conjunction with one specific UNDEFINED GROUP have to be allocated in the same PCU (i.e. the call to FC 95 defining the status of an undefined Group and all MePos using this UNDEFINED GROUP) ! All UNDEFINED GROUP processing is done within the AbvSPU Signal Processing Unit of the respective PCU.

With n PCUs in one IMAC L configuration it is theoretically possible to use the same UNDEFINED GROUP number in each of the n PCUs for a different purpose. This may nevertheless be very confusing for operators or during engineering. Therefore it is strongly recommended to use each UNDEFINED GROUP number just once in a system in one specific PCU.

## 9.2 Signal Output Groups

IMAC L will use SIGNAL OUTPUT GROUPS to indicate that one (or more) MePos now have a specific condition that is to be indicated / evaluated. This function can be used for any kind of indication or control reaction to this condition (e.g. for lamp indications).

One unique SIGNAL OUTPUT GROUP number is used to identify an IMAC L SIGNAL OUTPUT GROUP. This SIGNAL OUTPUT GROUP number has to be referenced to by one or more MePos in the MePo list, which are meant to control this SIGNAL OUTPUT GROUP.

The switching of the SIGNAL OUTPUT GROUPS condition is done depending on the different statuses of the MePo(s) controlling a SIGNAL OUTPUT GROUP. A SIGNAL OUTPUT GROUP will be active (TRUE) as long as at least one MePo in a PCU has a status, that has this SIGNAL OUTPUT GROUP assigned to it [→ 5.3.35 Fields : 'SOG' on page 54]. A SIGNAL OUTPUT GROUP will be inactive (FALSE), if no MePo in a PCU has a status, which has this SIGNAL OUTPUT GROUP assigned to it.

IMAC L itself will only evaluate and store the current conditions of all SIGNAL OUTPUT GROUPS. No code is being generated to further evaluate or use this information. The user program has to request the status of all required SIGNAL OUTPUT GROUPS by performing a simple call to FC 93 ("IMAC\_SOGGrp\_GET", one call per SOGrp) :

```
//--- Signal Output Groups Evaluation -----  
CALL "IMAC_SOGGrp_GET"  
SO_GRP :=1 // Get status of Signal Output Group 0  
SO_STATUS:=A2.0 // Use Signal Output Group to control 'what ever is desired'  
//-----
```

The parameters of FC93 "IMAC SOGrp GET" are :

### SO\_GRP:

Number of the Signal Output Group, for which the status is to be detected. 'SO\_GRP' is of type integer. The range of valid inputs for 'SO\_GRP' is 1...255 for the Signal Output Groups existing in IMAC L.

### SO STATUS:

The Boolean output parameter 'SO\_STATUS' contains, as a return value, the current status of this Signal Output Group : 'Active' [TRUE] or 'Inactive' [FALSE].

In the example above the call to FC 93 will detect the status of SIGNAL OUTPUT GROUP No. 1 and will copy the detected result directly output 'A1.0' . Output A1.0 will be TRUE, if this SIGNAL OUTPUT GROUP is active and FALSE, if this SIGNAL OUTPUT GROUP is inactive.

In IMAC L the HMI graphics display 'XSO' can be used to monitor, which status of which MePo has caused which SIGNAL OUTPUT GROUP to be active (filtering for SOGrp is possible).

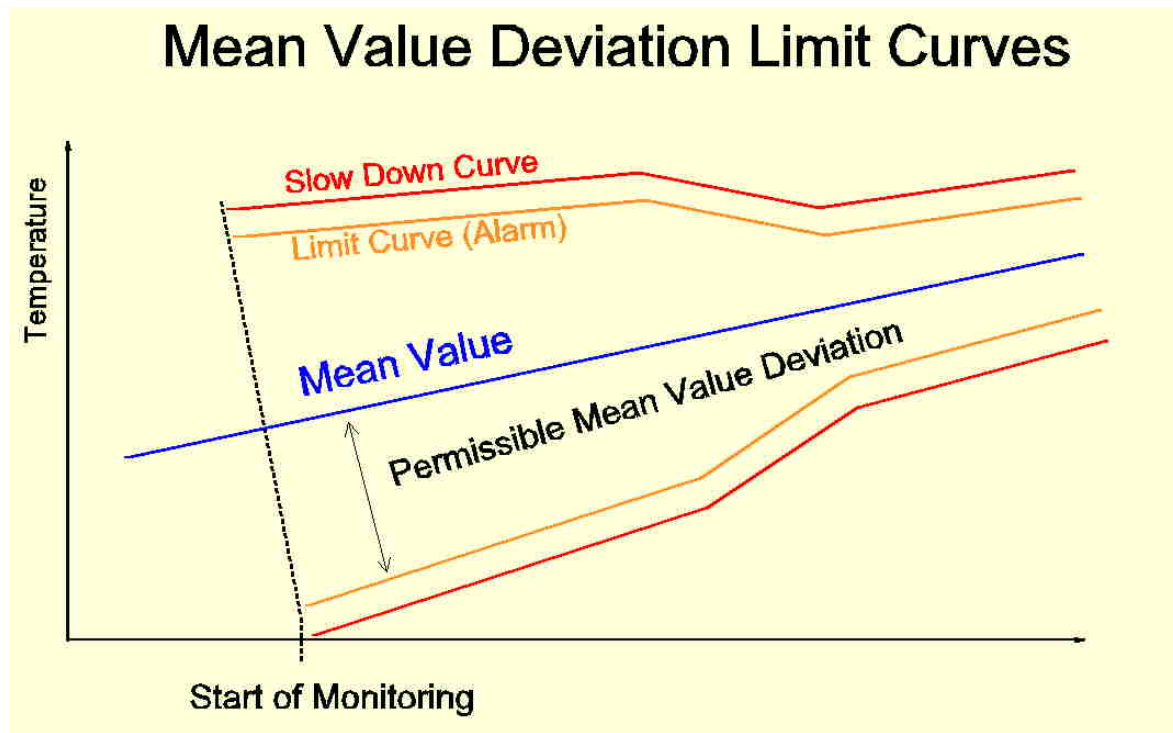
**Important Restriction :** For IMAC L the MePo(s) controlling one specific SIGNAL OUTPUT GROUP have to be allocated in the same PCU as the call to FC 93 detecting the status of this undefined Group ! All UNDEFINED GROUP processing is done within the AbvSPU Signal Processing Unit of the respective PCU.

With n PCUs in one IMAC L configuration it is theoretically possible to use the same SIGNAL OUTPUT GROUP number in each of the n PCUs for a different purpose. This would nevertheless mess up the indications of OSs (especially display 'XSO'). Therefore it is required to restrict the use of each SIGNAL OUTPUT GROUP to one specific PCU.

### 9.3 Exhaust Gas Mean Value Deviation Computing

It is common practice on ships, to monitor the exhaust gas temperature of single diesel engine cylinders with respect to the mean value of all cylinders of this engine. This gives a good indication of the performance of every single cylinder and a good pre-warning system for upcoming engine damages related to single cylinders. Variable (dynamic) limits are set to the computed deviations to achieve the desired monitoring function.

The following graph shows a typical example for type of exhaust gas deviation monitoring :



© Siemens AG 2009-10-01, Marine Engineering - IMAC 65 Ho - MvCo\_12.dxf Page 1/8

To achieve this function, a couple of computations have to be performed in addition to normal MePo engineering :

- Compute the exhaust gas temperature 'Mean value' of all valid cylinder temperatures. All temperature MePos being either 'Out of Range' or 'Undefined' or 'Faded Out' have to be taken out of this mean value calculation. Only the remaining (valid) cylinder temperature MePos may be used to calculate the mean value.
- Compute the deviation of every single cylinder temperature from this mean value. The results have to be stored in a data block and will be used to make up an internal 'Exhaust Gas Temperature Deviation' MePo for each cylinder.
- Apply a 'symmetrization' offset for each cylinder, if cylinder symmetrization is switched on. This offset is meant to compensate for deviations resulting from manufacturing tolerances between cylinders (there are always 'busy' and 'lazy' cylinders within one engine). These tolerances are normal and not subject of the deviation monitoring.

The following sections will describe how to set up this computation. Basically the engineering is reduced to filling one data block with parameters and use the computed result placed in a second data block.

### 9.3.1 IMAC L Exhaust Gas Temperature Deviation Function

All of the required computations are performed in the IMAC L exhaust gas computing function 'FC 99 IMAC\_EXHAUST'. This function will as a default be called cyclically in every OB1 cycle and will not require any parameters for calling this function nor any modification to the function itself :

```
//--- Technology Program -----
//CALL FBxxx,DBxxx
CALL "IMAC_EXHAUST"
//-----
```

All parameter settings are done in DB98. All input and output data are in DB99.

### 9.3.2 Parameters required to set up Exhaust Gas Function

All parameters required for this computation have to be entered to the data block 'DB98'. The calculations can be performed for any number of engines. Each engine may have up to 24 cylinder temperatures. 'DB 98' will be used to set these engine and cylinder numbers and to specify the references to all cylinder temperature MePos used. All data in this DB are set during the engineering.

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	NO_OF_ENGINES	BYTE	B#16#2
+2.0	ENGINE_1	STRUCT	
+0.0	IN_SYM_MAX_LIMIT	REAL	3.000000e+001
+4.0	IN_NO_OF_CYL	INT	10
+6.0	IN_OBJ_NO_01	INT	1535
+8.0	IN_OBJ_NO_02	INT	1537
+10.0	IN_OBJ_NO_03	INT	1539
+12.0	IN_OBJ_NO_04	INT	1541
+14.0	IN_OBJ_NO_05	INT	1543
+16.0	IN_OBJ_NO_06	INT	1545
+18.0	IN_OBJ_NO_07	INT	1547
+20.0	IN_OBJ_NO_08	INT	1549
+22.0	IN_OBJ_NO_09	INT	1551
+24.0	IN_OBJ_NO_10	INT	1555
+26.0	IN_OBJ_NO_11	INT	0
+28.0	IN_OBJ_NO_12	INT	0
+30.0	IN_OBJ_NO_13	INT	0
+32.0	IN_OBJ_NO_14	INT	0
+34.0	IN_OBJ_NO_15	INT	0
+36.0	IN_OBJ_NO_16	INT	0
+38.0	IN_OBJ_NO_17	INT	0
+40.0	IN_OBJ_NO_18	INT	0
+42.0	IN_OBJ_NO_19	INT	0
+44.0	IN_OBJ_NO_20	INT	0
+46.0	IN_OBJ_NO_21	INT	0
+48.0	IN_OBJ_NO_22	INT	0
+50.0	IN_OBJ_NO_23	INT	0
+52.0	IN_OBJ_NO_24	INT	0
+54.0	Spare	ARRAY[1..46]	
*1.0		BYTE	
=100.0		END_STRUCT	
+102.0	ENGINE_2	STRUCT	
+0.0	IN_SYM_MAX_LIMIT	REAL	3.000000e+001
+4.0	IN_NO_OF_CYL	INT	0
+6.0	IN_OBJ_NO_01	INT	0



In the first cell (DB98.DBW0 '**NO\_OF\_ENGINES**') the total number of engines has to be specified, that have to be computed. This number can be set to zero, if no exhaust gas computing is required. Please be careful with engine numbers > 9. The numerical representation of byte numbers in DBs is always hexadecimal! Using 10 engines will require the input value 'B#16#A'.

Starting from DB98.DBW2 one '**ENGINE\_x**' structure has to be placed, that contains all necessary parameters for one engine. This structure has a fixed length of 100 bytes. The default length of the whole DB98 is defined for use with 2 engines. If more than two engines are required, the DB may be extended to the required size by appending and filling in the '**ENGINE\_x**' data structure as many times as required.

The first parameter in each '**ENGINE\_x**' structure (offset +0.0) is named **IN\_SY\_MAX\_LIMIT**. This parameter is of type 'REAL' and defines the maximum permissible symmetrization offset value allowed. Any offsets higher than this value will be limited to this value. This limitation is a kind of a precaution to avoid setting offset values, which are way-off from reality. If no special requirements are specified, leave this set to a value of 30.0 (having degrees Celsius in mind). Modify accordingly, if temperatures are in any other unit like degrees Fahrenheit.

The second parameter in each '**ENGINE\_x**' structure (offset +4.0) is named **IN\_NO\_OF\_CYL**. This parameter is of type 'INT' and defines the number of cylinder temperatures to be evaluated for this engine. This parameter may be set to any value between 0 and 24. Setting this value to zero will skip this engine completely in computing. Values higher than 24 will be internally limited to 24. The number of cylinders specified here has to match the references to the cylinder exhaust gas MePos specified in the next cells of the structure !

The next 24 parameters in each '**ENGINE\_x**' structure (offset +6.0 to +52.0) are named **IN\_OBJ\_NO\_xx** (where xx is the cylinder number). These parameters are of type 'INT' and define the IMAC L object number of the MePo that is used to measure the exhaust gas temperature of the respective cylinder. For engines with 'n' cylinders only the first n parameters (out of the 24 possible) have to be filled in. The remaining (24-n) parameters are not evaluated and should be set to zero.

**HINT:** The easiest way to transfer the required object numbers into this DB is the new IMAC L function as described in section '9.7.1 Automatic Retrieval of IMAC L Object Numbers'. This function will automatically determine the required object numbers from MePo PointID names (strings variables placed in a DB) . The Exhaust Gas computing contained in the IMAC L template project uses precisely this method. Please see the a.m. section and the template project for more information on this topic. In DB98 enter '0' for all object numbers which are retrieved using this method.

**HINT:** Please remember (after modifying values in the declaration view) to initialize the data block in its 'Data View' (View → Data View + Edit → Initialize Data Block).

### 9.3.3 I/O data of Exhaust Gas Function

All input and output data for exhaust gas computing are allocated in DB99.

Starting from DB99.DBW0 one '**ENGINE\_x**' structure has to be placed, that contains all necessary input & outputs for one engine. This structure has a fixed length of 250 bytes. The default length of the whole DB99 is defined for use with 2 engines. If more than two engines are required, the DB may be extended to the required size by appending and filling in the '**ENGINE\_x**' I/O data structure as many times as required.

The first parameter in each '**ENGINE\_x**' structure (offset +0.0) is named **IN\_SYM\_CMD**. This parameter is of type 'BOOL' and is the basic input command bit, which will toggle the symmetrization



ON/OFF. Every time a 'toggle' is desired, this command bit has to be set to TRUE. The IMAC function will reset this bit, after the toggle has been performed.

The second parameter in each 'ENGINE\_x' structure (offset +0.1) is named **OUT\_SYM\_STATUS**. This parameter is of type 'BOOL' and is the output bit, which will indicate the current symmetrization status ON/OFF. Do not write / modify this bit (it is used to internally store the symmetrization status)!

The third parameter in each 'ENGINE\_x' structure (offset +2.0) is named **OUT\_MEAN\_VALUE**. This parameter is of type 'REAL' and contains the computed mean value for this engine.

The next 24 parameters in each 'ENGINE\_x' structure (offset +6.0 to +98.0) are named **OUT\_DEV\_CYLxx** (where xx is the cylinder number). These parameters are of type 'REAL' and contain the computed exhaust gas temperature deviation value of each cylinder. For engines with 'n' cylinders only the first n parameters (out of the 24 possible) are computed. The remaining (24-n) parameters are not written / modified. These values are commonly used as the data source for n internal MePos representing the 'Exh Gas Temp Dev Cyl. N'.

The last 24 parameters in each 'ENGINE\_x' structure (offset +102.0 to +194.0) are named **OUT\_SYM\_OFFS\_xx** (where xx is the cylinder number). These parameters are of type 'REAL' and contain the currently active symmetrization offset value of each cylinder. For engines with 'n' cylinders only the first n parameters (out of the 24 possible) are computed. If the symmetrization is OFF, all these offsets will be set to zero. The remaining (24-n) parameters are not written / modified. Do not write / modify these values, because these cells are used to internally store the active symmetrization offset values. These values may be read / indicated in any MePo / graphics picture, if they should be of any interest.

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	ENGINE_1	STRUCT	
+0.0	IN_SYM_CMD	BOOL	FALSE
+0.1	OUT_SYM_STATUS	BOOL	FALSE
+2.0	OUT_MEANVALUE	REAL	0.000000e+000
+6.0	OUT_DEV_CYL01	REAL	0.000000e+000
+10.0	OUT_DEV_CYL02	REAL	0.000000e+000
+14.0	OUT_DEV_CYL03	REAL	0.000000e+000
+18.0	OUT_DEV_CYL04	REAL	0.000000e+000
+22.0	OUT_DEV_CYL05	REAL	0.000000e+000
+26.0	OUT_DEV_CYL06	REAL	0.000000e+000
+30.0	OUT_DEV_CYL07	REAL	0.000000e+000
+34.0	OUT_DEV_CYL08	REAL	0.000000e+000
+38.0	OUT_DEV_CYL09	REAL	0.000000e+000
+42.0	OUT_DEV_CYL10	REAL	0.000000e+000
+46.0	OUT_DEV_CYL11	REAL	0.000000e+000
+50.0	OUT_DEV_CYL12	REAL	0.000000e+000
+54.0	OUT_DEV_CYL13	REAL	0.000000e+000
+58.0	OUT_DEV_CYL14	REAL	0.000000e+000
+62.0	OUT_DEV_CYL15	REAL	0.000000e+000
+66.0	OUT_DEV_CYL16	REAL	0.000000e+000
+70.0	OUT_DEV_CYL17	REAL	0.000000e+000
+74.0	OUT_DEV_CYL18	REAL	0.000000e+000
+78.0	OUT_DEV_CYL19	REAL	0.000000e+000
+82.0	OUT_DEV_CYL20	REAL	0.000000e+000
+86.0	OUT_DEV_CYL21	REAL	0.000000e+000
+90.0	OUT_DEV_CYL22	REAL	0.000000e+000
+94.0	OUT_DEV_CYL23	REAL	0.000000e+000
+98.0	OUT_DEV_CYL24	REAL	0.000000e+000
+102.0	DATA_SYM_OFFS_01	REAL	0.000000e+000
+106.0	DATA_SYM_OFFS_02	REAL	0.000000e+000
+110.0	DATA_SYM_OFFS_03	REAL	0.000000e+000

### 9.3.4 Control of Exhaust Gas Symmetrization

The status ON/OFF of the exhaust gas symmetrization function can be toggled and indicated with the two bits in DB 99 as described in the previous section. Usually there will be a button in one of the exhaust gas graphics pictures. This button will have a 'job' assigned to it, which will set a MePo of type 'PV' (process variable) to TRUE. This PV will point to the respective **IN\_SYM\_CMD** bit in DB 99. Every time this button is pressed, the status will be toggled between ON and OFF.

The **OUT\_SYM\_STATUS** will indicate the current ON/OFF status and can be used for indication. If the symmetrization is being switched OFF, all offset values will simply be set to zero. This causes the symmetrization to be OFF as a result.

If the symmetrization is being switched ON, all offset values will be set to the negative value of the current deviation of each cylinder. Switching this function to ON contains the implicit command from the operator, that the engine is in normal operating conditions now and that the current deviation temperature 'image' of the engine is its 'natural' status as it results from manufacturing tolerances. In the immediate moment after switching the symmetrization ON, the deviation values of all cylinders will go to zero, because they are compensated with the negative offsets taken from the engine in this moment. The offset values as taken from the engine are stored in DB 99. From now on, only deviations from this current status will be monitored and alarmed. By doing this, the influence from manufacturing tolerances is taken out of this monitoring.

Inadvertent setting of the symmetrization to ON where the engine is NOT in normal operating conditions (e.g. while a cylinder is damaged) has to be avoided. Basically this is deemed to be user responsibility. It is nevertheless recommended to assign user rights to the operation of the respective button, which makes the toggle function accessible to a Chief Engineer only ! Another protection against inadvertent setting of 'way-off' offset values is the limitation to a maximum offset amount (DB98, parameter **IN\_SYM\_MAX\_LIMIT**).

### 9.3.5 Measuring Points required to set up Exhaust Gas Deviation Alarms

The following MePos will be required to set up a typical 'Exhaust Gas Mean Value Deviation' monitoring System :

#### **Basic Cylinder Temperatures:**

Standard MePos to measure the temperatures of single cylinders. Typically these are of NiCrNi type. Fixed limits may be used to alarm absolute deviations of a single cylinder. Use limit values as specified by the engine manufacturer.

#### **Cylinder Temperature Mean Value:**

This value is computed as a result of FC99 and is stored as a real value in DB99.ENGINE\_x.OUT\_MEANVALUE. A MePo has to be set up pointing to this cell to make the mean value available for indication and alarming. This MePo typically needs one limit, which is used to control the BLOCKING GROUP for 'Low load' conditions of the engine. Use limit values as specified by the engine manufacturer.

#### **Cylinder Temperature Deviations from Mean Value:**

MePos for deviation of a single cylinder from the computed mean value. These MePos are used for indication and alarming and are stored as a real value in DB99.ENGINE\_x.OUT\_DEV\_CYLy. Typically these MePos will have a set of dynamic limits (2 or 4) which will cause the essential 'Mean Value Deviation Alarms'. Use limit values as specified by the engine manufacturer. The MePo controlling the respective DYNAMIC LIMIT GROUPS is the 'Cylinder Temperature Mean Value' MePo. The alarms resulting from these dynamic limits will be blocked during 'low load' conditions. The controlling MePo for this

Blocking Group is again the 'Cylinder Temperature Mean Value' MePo. The deviation alarms will be blocked, if the mean value is below its defined 'low load' limit.

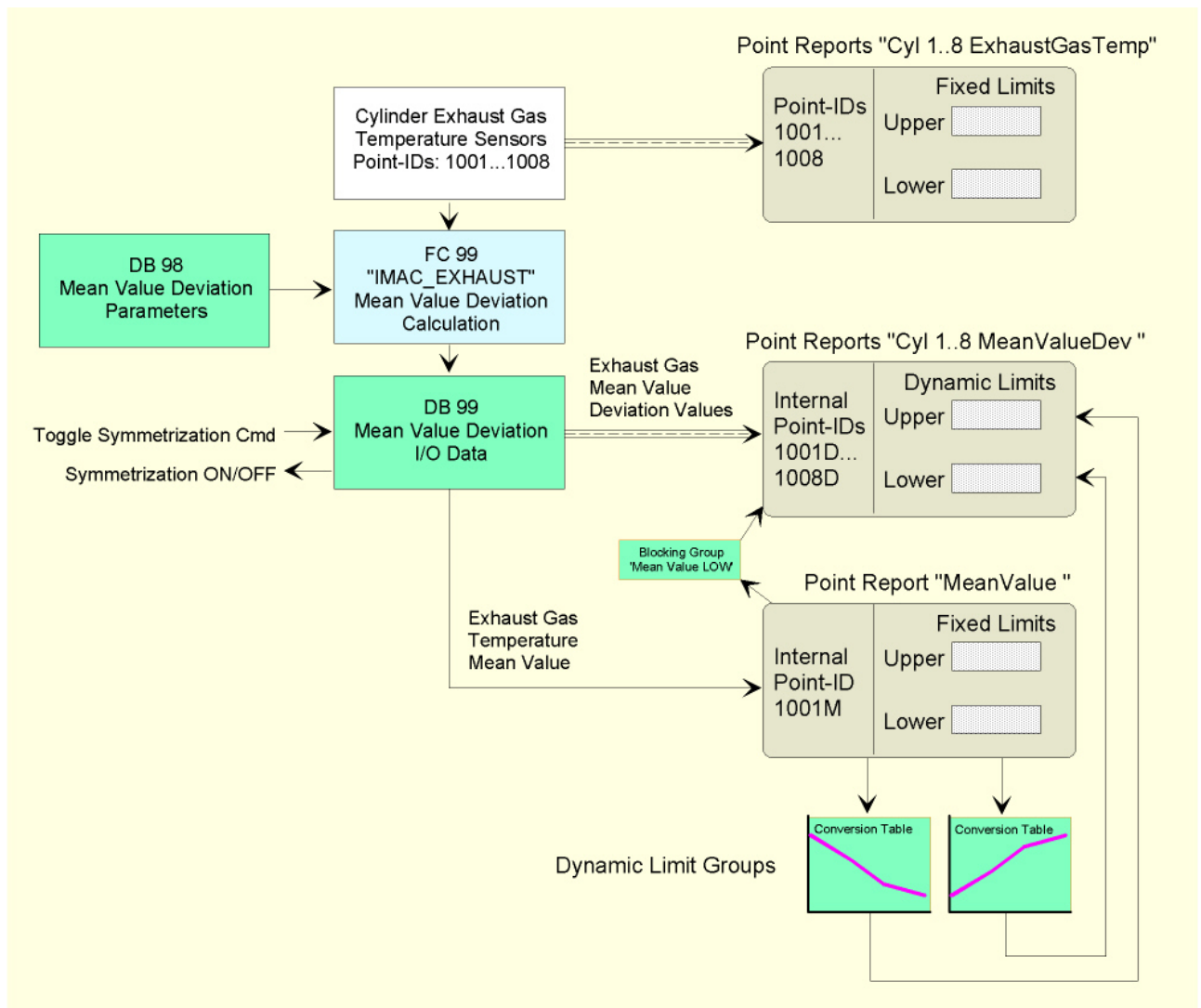
### Toggle Symmetrization Command:

Process variable (PV) of binary type. Every time this PV is being set to 'TRUE' the symmetrization will be toggled between 'ON' and 'OFF'. This bit will be reset by FC 99 after executing the command.

### Symmentrization Status:

Feedback from FC99 to indicate, if the symmetrization is currently 'ON' (TRUE) or 'OFF' (FALSE).

The following figure shows the basic dependencies between these MePos (Point IDs are examples only):



## 9.4 Monitoring of Profibus DP Slaves

IMAC L PCUs are, as a standard, set up to continue their program execution, even if one or more of the Profibus I/O units are missing. Due to this the fault(s) of Profibus slaves have to be monitored.

The OB1 contains the call(s) to a diagnostic function FC 95 "IMAC\_DP\_DIAG". Every instance of this function will collect all Profibus Slave Status information for one DP master system.

The working data and results of this diagnostic function are stored in DB 95 'IMAC\_DP\_DIAGN' in a data block of 100 bytes per master system. DB 95 is prepared for up to 4 master systems. For more master systems, DB95 has to be extended by 100 bytes per master system (copy & paste existing block).

If more than one master system is used, the call to FC95 has to be duplicated & modified for the other master systems, using an individual data set in DB95 for each call.

A typical Profibus diagnostic function call will look as follows (for the first master system) :

```
//--- Diagnostic Infos on Profibus DP Master System 1 -----
CALL "IMAC_DP_DIAG"
  CHECK_ACTIV          :=TRUE
  EXTERNAL_DP_INTERFACE :=FALSE // False for CPU-builtin DP-Interface
  DP_MASTERSYSTEM      :=1      // Master System Number
  DATA_FIELD          :="IMAC_DP_DIAGN".DATA_FIELD1
  SUM_SLAVES_DIAG      :="IMAC_DP_DIAGN".SUM_SLAVES_DIAG1
  LIST_SLAVES_NOT_PRESENT :="IMAC_DP_DIAGN".LIST_SLAVES_NOT_PRESENT1
  LIST_SLAVES_ERROR     :="IMAC_DP_DIAGN".LIST_SLAVES_ERROR1
  RETVAL               :="IMAC_DP_DIAGN".RETVAL1
  BUSY                 :="IMAC_DP_DIAGN".BUSY1
```

The parameters of FC95 "IMAC\_DP\_DIAG" are :

### **CHECK ACTIVE:**

Run flag for diagnosis. Always TRUE for performing this function in every cycle. For very big Profibus systems this flag may be used to create a time-sharing between various master systems by calling only one of the FC95 instances per cycle.

### **EXTERNAL DP INTERFACE:**

TRUE for 'external DP-bus interfaces (CP). FALSE for CPU-built-in DP-bus interfaces.

### **DP\_MASTERSYSTEM:**

Number of the Profibus-Mastersystem to be processed.

### **DATA FIELD / SUM SLAVES DIAG / LIST SLAVES NOT PRESENT / LIST SLAVES ERROR / RETVAL / BUSY:**

Pointers to the data fields of this master System in DB 95. Modify to point to the same field within another master system for adapting to other master systems.

The results of the FC 95 function calls are stored in the DB95 data Block as two bit arrays. For every possible PB slave there is one bit indicating, if the respective slave is missing completely (LIST\_SLAVES\_NOT\_PRESENT) and one bit indicating, if the respective slave currently has an error condition (LIST\_SLAVES\_ERROR).

The diagnostic result of each Profibus slave in a configuration will be used to set one UNDEFINED GROUP per PB slave using the FC96 "IMAC\_DP\_UNDEF". For a standard PCU (with 2 rows of ET200S peripherals) the setting of the UNDEFINED GROUPS will be as follows:

```
//--- Make Undefined Groups out of DP Diagnostic Infos -----
CALL "IMAC_DP_UNDEF"
  PB_DIAG_DB   := "IMAC_DP_DIAGN" // Diagnostic DB for DP-Bus Master System #1
  PB_DIAG_DW   := 0               // Offset Address for this master system
  PB_SLAVE_ADDR:= 1               // Profibus Slave Address ET200S No. 1
  UNDEF_GRP    := 1               // Undefined Group assigned to this PB Slave
  SLAVE_MISSING:= "IMAC_DP_FAULTS"._PB_slave_missing_01 // Diagnostic Status of PB Slave 1 (TRUE = FAULT)
  SLAVE_ERROR  := "IMAC_DP_FAULTS"._PB_slave_error_01  // Diagnostic Status of PB Slave 1 (TRUE = FAULT)

CALL "IMAC_DP_UNDEF"
  PB_DIAG_DB   := "IMAC_DP_DIAGN" // Diagnostic DB for DP-Bus Master System #1
  PB_DIAG_DW   := 0               // Offset Address for this master system
  PB_SLAVE_ADDR:= 2               // Profibus Slave Address ET200S No. 2
  UNDEF_GRP    := 2               // Undefined Group assigned to this PB Slave
  SLAVE_MISSING:= "IMAC_DP_FAULTS"._PB_slave_missing_02 // Diagnostic Status of PB Slave 1 (TRUE = FAULT)
  SLAVE_ERROR  := "IMAC_DP_FAULTS"._PB_slave_error_02  // Diagnostic Status of PB Slave 1 (TRUE = FAULT)
```

The parameters of FC96 "IMAC\_DP\_UNDEF" are :

**PB DIAG DB:**

Pointer to the DB used for storage of Diagnostic information. In IMAC L this is always DB 95.

**PB DIAG DW:**

Start address of the block to be used for this master system (in bytes).

**PB SLAVE ADDR:**

Profibus address of the slave to be monitored.

**UNDEF\_GRP:**

Number of the IMAC L UNDEFINED GROUP used to monitor this slave.

**SLAVE MISSING:**

Output bit indicating, that this slave is currently missing. Slaves can only be missing, if they are in the Simatic Manager hardware configuration. This bit can be used to create a MePo indicating and alarming the 'Slave missing' condition. As a default this bit is stored to DB94. Creating a binary MePo that evaluates this bit is strongly recommended.

**SLAVE ERROR:**

Output bit indicating, that this slave currently has an error condition (but may still be working). Slaves can only be faulty, if they are in the Simatic Manager hardware configuration. This bit can be used to create a MePo indicating and alarming the 'Slave error' condition. As a default this bit is stored to DB94. Creating a binary MePo that evaluates this bit is strongly recommended.



## 9.5 Audible Alarm and 'Horn Stop'

IMAC L as an alarm system will require to actuate one or more AUDIBLE ALARM signals in order to notify operators of incoming new events. At least one AUDIBLE ALARM output signal is required for each AUDIBLE ALARM GROUP [→ 5.3.34 Fields : 'AAG' on page 53] of a system.

The OPERATOR STATIONS will monitor for incoming events in any of the up to seven AUDIBLE ALARM GROUPS. Every MePo encountering a change in its Alarm Level from lower to higher values will trigger the AUDIBLE ALARM of the AUDIBLE ALARM GROUP assigned to this new (higher) alarm level. If such an event is detected, the OSs will set two PVs (Process Variables) to actuate the AUDIBLE ALARM of the respective AUDIBLE ALARM GROUP. One of these PVs is located in PCU 1 and the other in PCU 2. Every AUDIBLE ALARM GROUP has such a set of two PVs assigned to it.

The actual control of the required AUDIBLE ALARM outputs is done via two PCUs (PCU 1 and PCU 2). Both PCU's will receive the same program to control the inputs and outputs required for AUDIBLE ALARMS. The output contacts from PCU1 and PCU2 to control the horn of an Audible Alarm Group will have to be wired in series. This results in a redundancy, so that no single fault will prevent audible alarming.

The hardwired Horn Stop, if used in a project, has to be wired to PCU 1 and 2 in parallel (to silence the horn in both PCUs at the same time).

A typical example of the program for (here with two AUDIBLE ALARM GROUPS) will look like this :

```
//--- Control of Audible Alarm Groups -----
U   E   1.0           // Horn Off Command from Local Input (edge sensitive)
=   "IMAC_AAGrp".AAGrp_x_HornOff_Loc_Cmd    // Horn off for all AAGR !

CALL "IMAC_AAGrp_Ctrl"           // Control Program for Audible Alarm Groups

U   "IMAC_AAGrp".AAGrp_1_Horn_Flag
=   A   1.0           // Output to control Horn of AAGR 1
U   "IMAC_AAGrp".AAGrp_2_Horn_Flag
=   A   1.1           // Output to control Horn of AAGR 2
// U   "IMAC_AAGrp".AAGrp_3_Horn_Flag
// =   A   x.y         // Output to control Horn of AAGR 3
// U   "IMAC_AAGrp".AAGrp_4_Horn_Flag
// =   A   x.y         // Output to control Horn of AAGR 4
// U   "IMAC_AAGrp".AAGrp_5_Horn_Flag
// =   A   x.y         // Output to control Horn of AAGR 5
// U   "IMAC_AAGrp".AAGrp_6_Horn_Flag
// =   A   x.y         // Output to control Horn of AAGR 6
// U   "IMAC_AAGrp".AAGrp_7_Horn_Flag
// =   A   x.y         // Output to control Horn of AAGR 7
//-----
```

In this example the input E1.0 is used for a local 'Audible Alarm Off' input (e.g. pushbutton in a console). This input is edge-sensitive (0→1 transition) and will silence the AUDIBLE ALARM of all AUDIBLE ALARM GROUPS. If this function is not required, these two program lines may be removed. Any other input instead of E1.0 may be used as desired.

The call to FC 97 "IMAC\_AAGRP\_CTRL" will handle all required functions for control of the required AUDIBLE ALARM GROUPS. No modifications are required.

The next part of the program will connect the computed output signals to the physical outputs used to actuate the signaling device(s). This section has to be modified to meet the required number of audible alarm groups and the correct signal outputs.



The standard type of these outputs is a relay output with 'normally closed' characteristics (TRUE = closed contact = no alarm, FALSE = open contact = alarm). The relay output contacts of PCU 1 and PCU 2 have to be connected in 'series' for every AUDIBLE ALARM GROUP in the system. Any of the two contacts opening will actuate the respective AUDIBLE ALARM GROUPS signaling device(s).

### 9.5.1 Alarm Forwarding

Typically it is required on ships to have an ALARM FORWARDING implemented for all AUDIBLE ALARM GROUPS. ALARM FORWARDING means, that if an AUDIBLE ALARM is not being silenced within a pre-defined time (e.g. 5 min), this alarm will be forwarded to the next step of bringing the situation to operators attention. Possible actions after expiry of the predefined time can be :

- Forwarding to another AUDIBLE ALARM GROUP (if the ship has more than one).
- Actuating an Alarm via the Extended Alarm System (EAS), if present
- Actuating of Engineer Alarm / Engineer Call System
- Actuating of General Alarm System

The specific reaction to an 'expired' AUDIBLE ALARM has to be agreed upon for each project. It is recommended to discuss this topic with the owner and with the classification society in charge.

The basic function of alarm forwarding can be easily achieved by creating one additional MePo 'Alarm Forwarding AAGrp x' of type 'internal binary' for every AUDIBLE ALARM GROUP. This MePo has its data source in DB95: "IMAC\_AAGrp".AAGrp\_x\_Horn\_Flag (x = AUDIBLE ALARM GROUP). These flags will indicate an active Audible Alarm with their 'FALSE' condition. By setting an alarm level to the 'FALSE' (active) condition of the 'Alarm Forwarding' MePo and by setting the 'Delay In' of this status to the desired 'forwarding delay time' the required forwarding can be easily implemented. The 'Delay Out' should be set to one second.

This 'Alarm Forwarding AAGrp x' MePo will usually be used in conjunction with a SIGNAL OUTPUT GROUP (SO), that will be active, if this situation occurs. See [→ 9.2 Signal Output Groups on page 107] on how to evaluate the condition of a SIGNAL OUTPUT GROUP. The reaction (forwarding) then has to be programmed as a reaction to the 'active' status of this MePo. The actual use of this boolean output signal for any type of forwarding is part of the project specific engineering (no code is being generated automatically for this purpose). It can e.g. be used to directly control an output that will initiate the desired reaction.

The forwarding is usually not required to be redundant. It can be implemented in PCU 1 or PCU 2 as desired.

## 9.6 Application Watchdog for OS Software-Tasks

In IMAC L the necessary software tasks on all Operator Stations will be monitored for continued operation. For this purpose each OS will have a set of three Process Variables (PVs), which will be cyclically incremented by the respective tasks. The standard time interval between increments from each task is 20 seconds. All these PVs are stored in PCU 1. In PCU 1 all these PVs are monitored for changes. If one of these PVs does not change for more than a predefined time (standard is 60 seconds), two reactions will follow as a standard:

- A binary alarm will be initiated (MePo is required in MePo-list, the IMAC L template files contains template points for all 8 OSs).
- One of the seven AUDIBLE ALARM GROUPS in IMAC will be actuated independently from any OS (default = AAGrp 1).

The PLC-resources used for this function are: FC89, FC91, DB91, T99. This 'Watchdog' function set is only available with Alpha-Vision versions dated February 2006 or later.

The figure below shows the standard call to perform this function (in OB1) :

```
//--- Monitoring of Operator Station Software Tasks -----
// Attention : This function uses *** Timer T99 *** !
// This function is only called for PCU 1. Check for Station Number:
L      "spuDB_CFG".D_54          // PCU Station Number (PCU1 = 1, PCU2 = 2 ...)
L      1
<>I
SPB    nmon                      // If not PCU 1: Bypass this function !

CALL   "IMAC_OS_MON_CALL"        // Monitor SW-Tasks of IMAC L OSs (Watchdog)
NO_OF_OS :=2                    // Number of OS to be monitored (min. 1)
OS_WITH_EAS :=0                 // Number of the OS that drives the EAS panels (0 = no EAS)
OS1_WITH_OS99:=TRUE             // Does OS1 run a SoftPLC7-Instance "OS99" (FALSE = NO, TRUE = YES)
MON_TIME :=60                   // Monitoring Time (delay before Alarm) in seconds
AAGRP :=1                       // Number of AAGrp-Horn to be directly actuated for new fail (0 = none)
*****
//
// *** ATTENTION !!! ***
// *** Please do not specify 'AAGRP = 0' without a good reason ! ***
// *** Doing so may seriously affect the operational safety ! ***
// *** of the IMAC L system ! Modify / Test with care ! ***
// *** If you specify AAGRP > 1, make sure that this AAGrp ***
// *** works properly (i.e. a Horn will sound for this AAGrp) ! ***
//
*****
nmon: NOP 0
```

This function will automatically be called from PCU 1 only.

The parameters of FC89 "IMAC OS MON CALL" are :

### NO OF OS:

Number of OSs to be monitored (minimum = 1, maximum = 8)

### OS WITH EAS:

Number of the OS that drives the EAS system ( 0 = no EAS System installed)

### OS1 WITH OS99:

Does OS1 run an instance of the 'OS99' SoftPlc7 program (FALSE = no, TRUE = yes) ?

### MON TIME:

Monitoring Time (in seconds). After expiry of this time without a reaction from all required OS tasks, the output reactions (binary alarm, audible alarm) will be initiated.

**AAGRP:**

AUDIBLE ALARM GROUP (AAGrp), from which the horn will be triggered to 'active', if a new alarm event has been detected for any of the OSs to be monitored. Standard Alarm processing requires the OSs to actuate the horn. This specific control function will directly access the AAGrp in PCU 1 (bypassing the OSs) and thus will work even if no OS is alive any more!

The required PVs are predefined for all 8 possible OSs in the AvEdit template project delivered with the IMAC L CD (the definition is to be found in '\_XlxPV.obj.xml'). The naming scheme for these PVs is as follows:

- WDGPRC0x for Task 'PRCS.EXE' (x = OS-number)
- WDGUDP0x for Task 'UDP2AV.EXE' (x = OS-number)
- WDGEAS0x for Task 'AV2SOFTEAS.EXE' (x = OS-number)
- WDGOS99 For SoftPlc7 'OS99' (running on OS1 only)

Customizing of this function (for up to 8 tasks, e.g. SoftPLC7 program instances) is possible by program modification. Customizing is then required in FC91. To achieve this, please refer to the extensive comments in FC91 and DB91 (no detailed explanation on this is available).

Each OS has to be set up to correctly control the required PVs. The necessary setting for the OSs has to be entered in the file 'C:\vision\bin\vision2000.ini'. The required section is named '[Watchdog]'. A typical example of this section should look like this:

```
[watchdog]
File=c:\vision\bin\wdog.txt
TouchTime=5000
;*****
; OS-specific Application Watchdog Settings
; OS-specific Naming of Variable : WDGPRC0x (where x = OS-Number) !
PrCs=WDGPRC01, 20
; OS-specific Naming of Variable : WDGUDP0x (where x = OS-Number) !
Udp2Av=WDGUDP01, 20
; OS-specific Naming of Variable : WDGEAS0x (where x = OS-Number) !
Av2SoftEas=WDGEAS01, 20
;*****
```

The example shown depicts the settings for OS 1 (naming of the PVs according to the a.m. scheme). The time setting for the desired frequency of incrementing the PVs by the monitored applications is in this example set to 20 (seconds) for all three applications. No entry is required for the 'OS99' program (fixed time at 20 seconds, directly in OB1 of this program).

The settings for other OSs will have to refer to the appropriate individual PVs of the OS in question instead. This means, that the watchdog function requires, that all OSs have different [watchdog] settings in the vision2000.ini file!

The IMAC L template files on the IMAC L CD contain 8 sample binary points named 16004 ... 16011. These points are set up to create one binary alarm for each of the OSs as a Watchdog failure alarm. The data source for these points is directly in PCU1 -> DB91. The required bits will be computed automatically by calling the a.m. function set.

**Important Remark :**

It is strongly recommended to use both methods of notification for a detected watchdog failure (one binary alarm for each OS AND direct control of an AAGrp in PCU1) ! The binary alarms will give detailed information on where to look for a malfunction, while the direct AAGrp control is kind of a fallback position, which will create an alarm even if all OSs should fail.

**Important Remark :**

The actuating of the AAGrp by the watchdog as specified in the call to FC89 will only be active in PCU 1 ! This means, that a correct alarming of AAGrpx for watchdog alarms will only be possible, if the output relay contacts for this AAGrp are correctly wired in parallel between PCU 1 and PCU 2 [→ 9.5 Audible Alarm and 'Horn Stop' on page 117]. This is a basic requirement for all AAGrp in IMAC L to achieve a redundancy in Alarm indication (e.g. in case that a PCU / output channel / fuse should fail).

## 9.7 Access to MePo Values and Status from Control Programs

In IMAC L control programs can access current MePo data (value, status), if the IMAC L object number is known. The next figure will show a typical example, on how a control program can access the 'Value' and 'Status' properties of an IMAC L MePo :

```
// Get Status of the Analogue MePo for this Cylinder
CALL "GetSts_ex"
obj :=#_ObjOfCurrentCylinder // Object Number of current Cylinder
wMask:=W#16#1F // Bit Mask for Significant Bits of Status (Bits 0-4)
wSts :=#_StatusOfCurrentCylinder // Status as a return Value
bOk :=#_ResultGetSts // Result Flag of FC166:GetSts_ex

// Get Value of the Analogue MePo for this Cylinder
CALL "GetValue_ex"
obj :=#_ObjOfCurrentCylinder
iVal:=#_i_dummy
rVal:=#_ValueOfCurrentCylinder // Value (Real) of MePo for Current Cylinder Temperature
bVal:=#_b_dummy
bOk :=#_ResultGetVal // Result Flag of FC167:GetVal_ex

// Compute the 'Fault' Flag fot this Cylinder
ON #_ResultGetSts // Fault in 'GetSts_ex' ?
ON #_ResultGetVal // Fault in 'GetVal_ex' ?
O{
L #_StatusOfCurrentCylinder // Status as a return Value
L 16 // AbvSPU Status 'Out of Range'
==I // Out of Range ?
}
O{
TAK
L 17 // AbvSPU Status 'Undefined'
==I // Undefined ?
}
O{
TAK
L 18 // AbvSPU Status 'Faded Out'
==I // Faded Out ?
}
= #_FaultCurrentCylinder // Fault of the temperature MePo of the current Cylinder
```

FC166 "GetSts\_ex" will return the status word of a MePo

The parameters of FC166 "GetSts\_ex" are :

**obj:**

Object number (from AvET) of the MePo in question. Only MePos having their 'home' in this PCU (where this function is called) can be requested.

**wMask:**

Mask to isolate a set of bits from this status. Usually the mask W#16#1F is used to get only the relevant bits (0...4). Coding of Status bits : see below.

**wSts:**

Status word of requested MePo (masked with mask wMask). For accessing the MePo value, it has first to be verified, that this value is valid (i.e. the MePo is not 'Out of Range' or 'Undefined' or 'Faded Out'). Coding of Status bits : see below.

**bOk:**

Result value. TRUE if function was successful. FALSE, if function failed (e.g. object with this number does not exist)

The IMAC L status word has the following coding :

Bits 5 – 15 as single Bits	Bits 0 – 4 coded as a binary Number
5: Error	1: MePo is in Status '1'
6:	2: MePo is in Status '2'
7: Unacknowledged Alarm	3: MePo is in Status '3'
8: Status '1' is blocked	4: MePo is in Status '4'
9: Status '2' is blocked	5: MePo is in Status '5'
10: Status '3' is blocked	6: MePo is in Status '6'
11: Status '4' is blocked	7: MePo is in Status '7'
12: Status '5' is blocked	8: MePo is in Status '8'
13: Status '6' is blocked	16: MePo is 'Out of Range'
14: Status '7' is blocked	17: MePo is 'Undefined'
15: Status '8' is blocked	18: MePo is 'Faded Out'

The parameters of FC167 "GetValue\_ex" are :

**obj:**

Object number (from AvET) of the MePo in question. Only MePos having their 'home' in this PCU (where this function is called) can be requested.

**iVal:**

Return Integer value. Used only, if MePo is of type 'Integer'.

**rVal:**

Return Real value. Used only, if MePo is of type 'Real'. All analogue MePos in IMAC L are of type 'Real'.

**bVal:**

Return Boolean value. Used only, if MePo is of type 'Boolean'.

**bOk:**

Result value. TRUE if function was successful. FALSE, if function failed (e.g. object with this number does not exist)

The return value from FC167 may only be used, if the evaluation of the respective MePos status word has shown, that this value is valid (i.e. the MePo is not 'Out of Range' or 'Undefined' or 'Faded Out'). An example for this evaluation is shown in the example above : In this example the result value '\_ValueOfCurrentCylinder' may only be used, if the flag '\_FaultCurrentCylinder' is FALSE.

**Important Hint :**

Please observe, that accessing this information by referencing object numbers makes it necessary, to take care of any changes of object numbers, as these will change if the engineering changes. It is strongly recommended use the method as described in section '9.7.1 Automatic Retrieval of IMAC L Object Numbers' below, to automatically retrieve IMAC L object numbers!



### 9.7.1 Automatic Retrieval of IMAC L Object Numbers

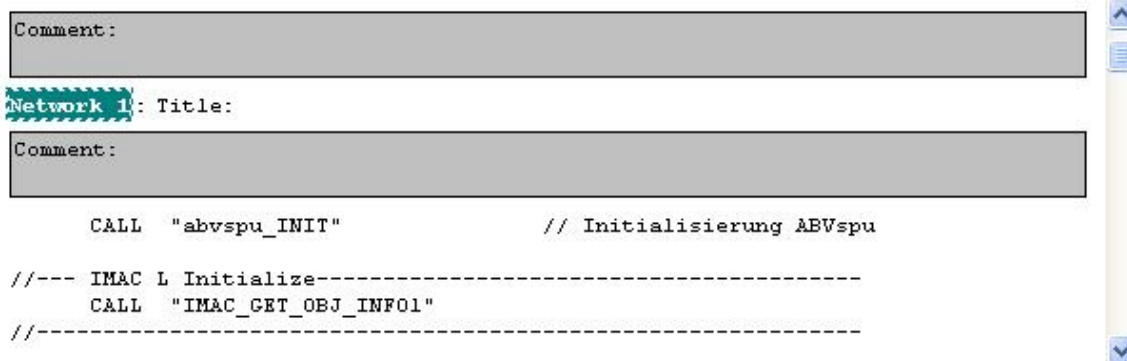
Object numbers in IMAC L will be fixed, if engineering is finished, but they will frequently change while engineering is under way. It is very likely that the object numbers will change within a PCU, if you add or remove MePos to / from this PCU during engineering.

Therefore it is at least difficult to refer to absolute object numbers, if you use the functions in section '9.7 Access to MePo Values and Status from Control Programs' for data access in PCUs. You will always have to check your program after each 'build' in IMAC L engineering and adjust object numbers which have changed. This is time consuming and prone to error.

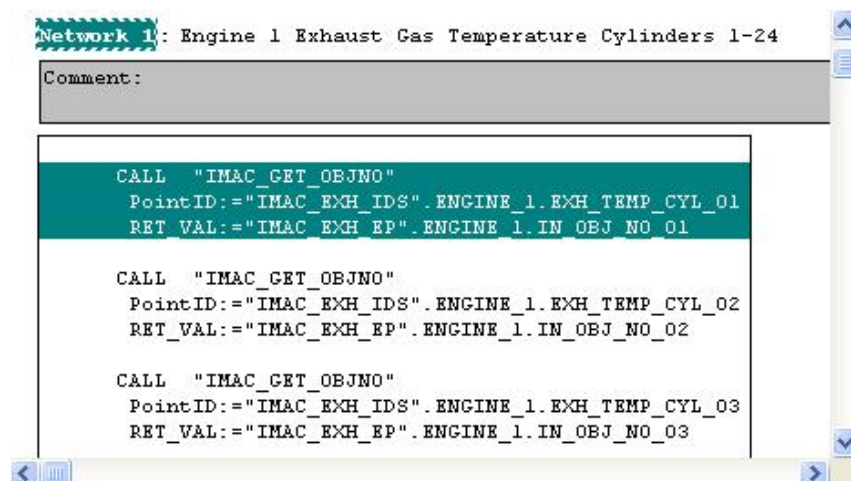
Therefore a new IMAC L function offers a possibility to automatically retrieve MePo object numbers during PLC startup. The only reference to the desired object number is the MePo Point ID. As this Point ID will only change in rare occasions, this is a very efficient and reliable method to retrieve object numbers and to refer to runtime data in an IMAC L PCU.

This object number retrieval will (if being performed for a significant number of MePos) be quite time consuming. Therefore IMAC L uses one single program call during PLC startup to retrieve and store all the required object numbers for one PCU.

The call to the 'IMAC L Object Number Retrieval' is located in OB100 (Startup of S7 300). The block used for collecting all required object numbers is **FC83 ('IMAC\_GET\_OBJ\_INFO1')** :



Within the FC83 there are several calls to **FC 85 ('IMAC\_GET\_OBJNO')** – one call for every object number that is required in the PLC program:



A call to FC 85 has only two parameters:

- **“Point ID”:**  
This is the IMAC L PointID of the MePo for which the object number is to be retrieved. This is a String variable, that has to be stored in a DB (FC’s will not accept ‘Strings’ as direct parameters in their call).
- **“RET\_VAL”:**  
This is the return value of type Integer, which is computed during the processing in IMAC\_GET\_OBJNO. The return value will be zero, if the MePo Point ID was not found in this PCU. If the return value is >0, the returned value is the desired object number.

A typical declaration of a ‘MePo String’ in a Simatic DB will look like this:

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	ENGINE_1	STRUCT		Number of Engines : See DB 98.NO_OF_ENGINES
+0.0	EXH_TEMP_CYL_01	STRING[12]	'030201'	Exhaust Gas TEMP Measuring Point ID from IMAC MePo list (String)
+14.0	EXH_TEMP_CYL_02	STRING[12]	'030202'	Exhaust Gas TEMP Measuring Point ID from IMAC MePo list (String)
+28.0	EXH_TEMP_CYL_03	STRING[12]	'030203'	Exhaust Gas TEMP Measuring Point ID from IMAC MePo list (String)

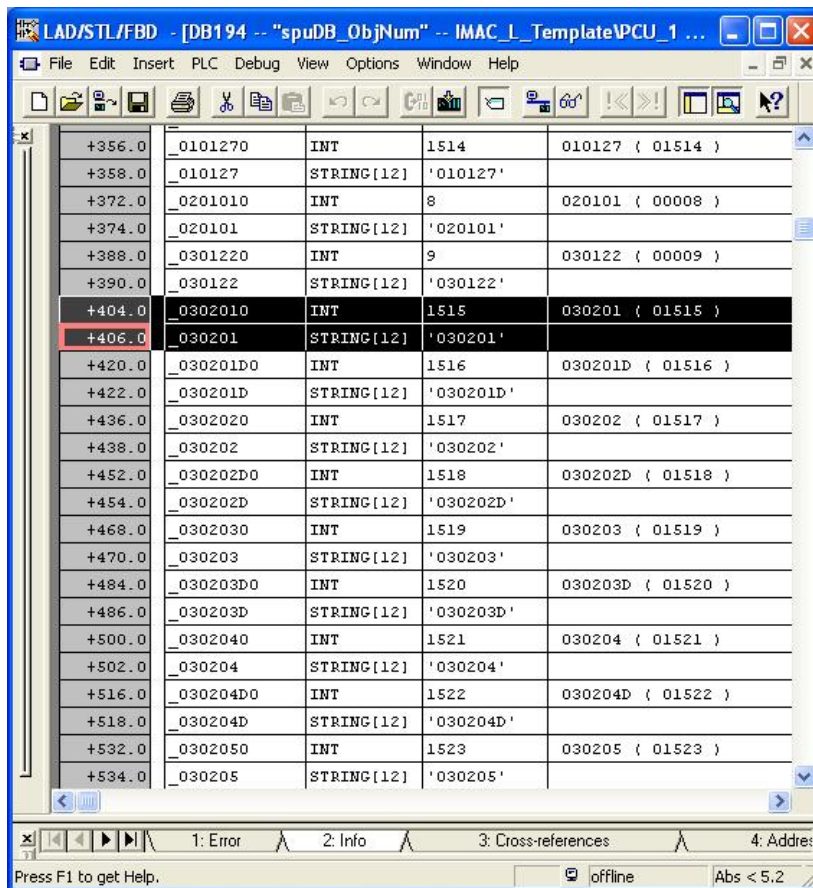
Please observe the following hints:

- The examples shown here are from the IMAC L Exhaust Gas Temperature computing (predefined in IMAC L template). You have to create the required DB entries for your own object number retrievals as required (e.g. use your own DB, modify FC83).
- Strings passed to FC85 have to be <= 12 characters in current length.
- The column ‘Initial Value’ contains the MePo PointIDs. Please remember to initialize such data blocks after each change. The commands required for this in the Simatic DB editor are : “View” → “Data View” & “Edit” → “Initialize Data Block”.
- If MePo PointIDs in your project use characters, please observe, that this object number retrieval is case-sensitive.

The retrieved IMAC L object numbers are usually stored in a DB location as well. This may be the same DB as for the ‘PointID’ parameter or a target DB address directly in your technology program. Here you see an example for IMAC L Exhaust Gas Temperature computing (actually all the screenshots in this section are taken from this Exhaust Gas template):

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	NO_OF_ENGINES	BYTE	B#16#1	Number of Engines to be processed (0 = no Exhaust Gas Calculations)
+2.0	ENGINE_1	STRUCT		
+0.0	IN_SYM_MAX_LIMIT	REAL	3.000000e+001	Input: Maximum Value (+/-) for Symmetrization Offset of every single
+4.0	IN_NO_OF_CYL	INT	10	Input: Number of Cylinders in this Engine (1...24, 0 = Off)
+6.0	IN_OBJ_NO_01	INT	0	Input: Object Number of IMAC MePo for Exhaust Gas Temperature Cylinde
+8.0	IN_OBJ_NO_02	INT	0	Input: Object Number of IMAC MePo for Exhaust Gas Temperature Cylinde

If you should want to look up the IMAC L object numbers, which are currently available in an IMAC L PCU, please open the **DB 'spuDB\_ObjNum' (DB194)**. This DB contains an (unsorted) list of pairs (object number / PointID). These are all PointIDs / Object Numbers which have been generated by IMAC L for this PCU. This in fact is the DB, from which the FC 85 'IMAC\_GET\_OBJNO' retrieves it's object number information. An Example of this DB would look like this:



Address	Object Number	Object Type	Object Value	PointID
+356.0	_0101270	INT	1514	010127 ( 01514 )
+358.0	_010127	STRING[12]	'010127'	
+372.0	_0201010	INT	8	020101 ( 00008 )
+374.0	_020101	STRING[12]	'020101'	
+388.0	_0301220	INT	9	030122 ( 00009 )
+390.0	_030122	STRING[12]	'030122'	
+404.0	_0302010	INT	1515	030201 ( 01515 )
+406.0	_030201	STRING[12]	'030201'	
+420.0	_030201D0	INT	1516	030201D ( 01516 )
+422.0	_030201D	STRING[12]	'030201D'	
+436.0	_0302020	INT	1517	030202 ( 01517 )
+438.0	_030202	STRING[12]	'030202'	
+452.0	_030202D0	INT	1518	030202D ( 01518 )
+454.0	_030202D	STRING[12]	'030202D'	
+468.0	_0302030	INT	1519	030203 ( 01519 )
+470.0	_030203	STRING[12]	'030203'	
+484.0	_030203D0	INT	1520	030203D ( 01520 )
+486.0	_030203D	STRING[12]	'030203D'	
+500.0	_0302040	INT	1521	030204 ( 01521 )
+502.0	_030204	STRING[12]	'030204'	
+516.0	_030204D0	INT	1522	030204D ( 01522 )
+518.0	_030204D	STRING[12]	'030204D'	
+532.0	_0302050	INT	1523	030205 ( 01523 )
+534.0	_030205	STRING[12]	'030205'	

## 9.8 Time Synchronization & Time Setting in IMAC L

IMAC L automatically synchronizes the time setting (local time (LT) and universal time (UT) between all PLCs and all OSs. For this purpose one of the PCUs will be the so called TIME MASTER. The default for the TIME MASTER is PCU 1. If the clocks of the other PCUs will deviate significantly between the IMAC L PCUs, the time settings of PCU 1 will automatically be used to set the clocks of those PCUs not being the TIME MASTER. Time settings are automatically transmitted to all connected OSs as well and will automatically cause the OSs to adjust the LT / UT time accordingly. If any PCU or OS has a significant deviation from the time setting of PCU 1, the time settings will be automatically synchronized to the time received from PCU 1.

IMAC L uses the UT settings to set the hardware clocks of the PLCs (PCUs) and PCs (OSs), while the LT settings are always held as an offset value (with respect to the UT) in a memory variable.

### 9.8.1. Setting of the IMAC L Universal Time (UT) from OS

The standard setting of UT is from the GD 'X2'. The user has to be logged in with sufficient user rights (e.g. Superuser 'SU'). If the user is not logged in with sufficient rights, the required buttons will be grayed and cannot be operated.

The part of GD 'X2' that is used to set the UT looks like this:

To set a new universal time and / or date, the operator has to proceed as follows:

- Press 'CTRL-F1' (user Login)
- Enter the user name (e.g. 'SU')
- Press Enter.
- Enter the user password
- Press Enter.  
(the buttons 'Set' & 'Abort' are now active)
- Click into the desired entry field. The default in the entry fields and in the command line is the current time / date of the OS. The default value from this field is copied to the command prompt.
- Enter / edit the value at the command prompt (not in the entry field).
- Press Enter.  
(The newly entered value will be displayed in the respective entry field, but is not yet active. Invalid input values will be rejected - i.e. will not show up in the entry field.)
- Click on the button 'Set'  
(after a short delay, the newly entered values will be active)
- Press 'ALT-F1' (user Logout)

The modified time and / or date values will be transmitted to the TIME MASTER-PCU after pressing the 'Set' button. The TIME MASTER will adjust its internal clock to the new value and then distribute the changes to all other PCUs and to all OSs in the system. In this moment the clocks of the PCUs and OSs will be set (including the OS from which the 'Set' command was issued).


The formats for entering the time / date are :

- Time : yy-mm-dd
- Date : hh:mm:ss

### 9.8.2. Setting of the IMAC L Local Time (LT) from OS

The standard setting of LT is from the GD 'X2'. The user has to be logged in with sufficient user rights (e.g. Superuser 'SU'). If the user is not logged in with sufficient rights, the required buttons will be grayed and cannot be operated.

The part of GD 'X2' that is used to set the LT looks like this:



To set a new local time LT, the operator has to proceed as follows:

- Press 'CTRL-F1' (user LogIn)
- Enter the user name (e.g. 'SU')
- Press Enter.
- Enter the user password
- Press Enter.  
(the buttons 'Set' & 'Abort' are now active)
- Click on the buttons +- 1 / 20 / 60 Minutes to adjust the local time  
(The newly entered value will be immediately set and distributed in the IMAC L system after each single click).
- Press 'ALT-F1' (user LogOut)

The modified time and / or date values will be transmitted to the TIME MASTER-PCU after each single click. The TIME MASTER will adjust its internal clock to the new LT value and then distribute the changes to all other PCUs and to all OSs in the system. In this moment the LT offsets of the OSs will be set (including the OS from which the 'Set' command was issued). This procedure causes a small time delay between a 'Click' and the resulting adjustment of the LT.

The LT in IMAC L is always stored as an 'offset' with respect to the UT. Therefore the LT date can only be set indirectly by adjusting the LT offset to a value that causes the resulting LT date to be 'yesterday' or 'tomorrow' (with respect to the UT).

LT settings with an offset of more than +- 12 hours do not make sense, having the purpose of this setting in mind. Offsets > +-12 hours are technically possible, but it is strongly recommended not to use them, as this may be confusing to the operator.

### 9.8.3. Setting of the IMAC L Universal Time (UT) from PCU

Setting of the IMAC L UT is only possible from the TIME MASTER PCU. Thus a ship's Master-Clock interface can only be connected to the TIME MASTER PCU !

Setting the PCU UT time is possible in two ways:

- Set the hardware clock of the PLC using the Simatic Manager's built-in function for this: Simatic Manager Menu 'PLC' → 'Diagnostic/Setting' → 'Set Time of Day'
- Calling the system FC 'SFC0 SET\_CLK' to set the PLC hardware clock to the UT. This is useful e.g. for connecting a master clock.

The resulting time will automatically be distributed in IMAC L and will be immediately active.



#### 9.8.4. Setting of the IMAC L Local Time (LT) from PCU

The local time offset in IMAC L is stored as a long integer variable (48 bit, signed) containing the time offset (in seconds) between the LT and UT. This 6 byte long LT offset value is stored in the following location :

- DB196.DBB90 (abvspu\_DB\_GLO.aLTC\_Correction[1]) LT Offset Bits 00 – 07; LSB
- DB196.DBB91 (abvspu\_DB\_GLO.aLTC\_Correction[2]) LT Offset Bits 08 - 15
- DB196.DBB92 (abvspu\_DB\_GLO.aLTC\_Correction[3]) LT Offset Bits 16 - 23
- DB196.DBB93 (abvspu\_DB\_GLO.aLTC\_Correction[4]) LT Offset Bits 24 - 31
- DB196.DBB94 (abvspu\_DB\_GLO.aLTC\_Correction[5]) LT Offset Bits 32 - 39
- DB196.DBB95 (abvspu\_DB\_GLO.aLTC\_Correction[6]) LT Offset Bits 40 – 47; MSB

Example: to set the LT offset to '-1 hour' it is necessary to write a value of -3600 seconds (hexadecimal = 'FF FF FF FF F1 F0') to the 6 bytes of LT offset. In a S7 program this will look like this:

```
//--- Example: Set LT offset to -1 hour (-3600 seconds) -----
L      B#16#F0                      // Store HEX value FF FF FF FF F1 F0 = -3600 (dec)
T      "abvspu_DB_GLO".aLTC_Correction[1]
L      B#16#F1
T      "abvspu_DB_GLO".aLTC_Correction[2]
L      B#16#FF
T      "abvspu_DB_GLO".aLTC_Correction[3]
T      "abvspu_DB_GLO".aLTC_Correction[4]
T      "abvspu_DB_GLO".aLTC_Correction[5]
T      "abvspu_DB_GLO".aLTC_Correction[6]
```

The LT may only be set in a Range of +/- 12 hours or +/- 43200 seconds. This is a numerical range of FFFFFFFF5740<sub>HEX</sub> to 00000000A8C0<sub>HEX</sub> for the aLTC\_Correction value. The upper four bytes of this value always have to be zero. Setting the local time outside of the permitted range may yield unexpected results (e.g. un-ergonomic display indications) and has to be avoided.

#### 9.8.5. Logging Time & Date setting Events in the EventLog

In IMAC L all time adjustments up to 19 minutes are assumed to be a 'clock adjustment' procedure and will not be logged. All time adjustments amounting to more than 19 minutes are assumed to be a 'clock setting' procedure and will be automatically logged in the IMAC L EVENTLOG.

A 'Changed Time' log entry will have the time stamp of the time 'before setting the time'. A text 'New LocalTime : yy-mm-dd hh:mm:ss' will indicate the time / date to which the setting has been made :

Point-Id	Description	Value	Unit	Status	Date UT	Time UT	Time LT
New	Local Time	06-02-15 13:52:16		Changed Time	06-02-15	13:52:16	12:52:16

#### 9.8.6. Consequences of Setting the UT time back into the Past

UT time adjustment is usually just necessary to take care of minor deviations between clocks in different systems of a ship. Therefore time setting of UT as a rule is just by some minutes forward or backward. If the IMACL time setting function is abused to set the UT time by a serious amount of time back into the past, a potential problem with the EVENTLOG display may arise from this abuse:

If the UT time is set 'backward' (i.e. into the past), the events being registered in IMAC L will be registered to have happened at this new time. Usually there are already events entered in the EVENTLOG resulting from events 'before setting the clock backward'. This makes these 'old' events



seem to lie in the future (from the point of view of the newly set time). As all EVENTLOG registration in IMAC L is displayed chronologically according to UT (newest on top), these future events will stay 'On Top' of the EVENTLOG list, until the new time has reached the time when these points have received their time stamp.

If the turning back of time is just for a few minutes or hours, this behavior does not do any serious harm, as it repairs itself within short time.

If the time has erroneously been set to a date far in the future and then back to the 'real' time, it may happen, that events that have happened in the future will stay on top of the EVENTLOG for a very long time and will possibly make it unusable (i.e. these events are staying permanently at the top of the list and prevent the user from easily recognizing recent events. Recent events are potentially only visible after scrolling down).

To repair this it is necessary to delete the EventLog databases of all OSs showing this symptom. The procedure below will remove all EventLog entries in the IMAC L system. All OSs to which this procedure is applied will start with an empty EVENTLOG database. Please inform / ask the Chief Engineer of a ship before doing so ! Important information may be lost and you may be held responsible for this data loss.

The procedure for deleting the EVENTLOG databases in an IMAC L OS is as follows:

- Stop all IMAC L applications on the OS
- Delete the two files 'C:\vision\bin\EVENTLOG.dbf' and 'C:\vision\bin\EVENTLOG.cdx'
- Empty the Windows 'Waste Bin' to free disk space
- Reboot the OS

#### 9.8.7. Time setting precision between Stations in IMAC L

The time setting algorithms in IMAC L are made up to set all the clocks in the system to the same time with the best possible precision. Nevertheless there are some restrictions to this precision:

- The time setting function of the Simatic S7 '300' family has a precision of  $\pm 1$  second (although the internal clock knows and shows milliseconds). Setting the 'milliseconds' part of the clock is not possible.
- The Simatic S7 '300' has no built-in time synchronization functions in its Ethernet CPs (like the S7-400 family has). Therefore the user program (here IMAC L) is responsible to distribute the time in the system. This is subject to bus-communication delays and processing delays that will reduce precision even more.
- All stations (except PCU 1) will cyclically receive and evaluate time telegrams from PCU1. To avoid permanent setting of all clocks (which would create all clocks to 'jitter' / appear to jump in their milliseconds), setting will only be performed, if the deviation is significant.

These restrictions cause the time synchronization in IMAC L to have an average precision of approximately  $\pm 5$  seconds between PCUs / OSs. This is a system property. Higher precision is not available. IMAC L is an alarm system and not a real-time analyzer.

### 9.8.8. Time Synchronization Strategy between Stations in IMAC L

IMAC L will automatically synchronize the time (LT and UT) and date between all active stations (OS & PCU). This is done in a way, that the TIMEMASTER PLC will cyclically distribute time synchronization telegrams on the bus. These telegrams will be received by all active stations in the system. Each station will compare its own clock with the values from the newly received telegram. If there is a significant deviation detected, a station will immediately set its own clock to the value given in the telegram.

The update time between the cyclical transmissions of two telegrams from the TIMEMASTER is adjusted in the TIMEMASTERS DB196.DBW14 (wTimeSyncCycleCount). This delay is measured in counts of OB1-cycles before the telegram is re-transmitted. The default is 600 cycles (0258<sub>HEX</sub>). For PCUs with large control programs and a long OB1 cycle time it is possible to adjust this value to smaller values.

### 9.8.8. TimeMaster Settings

The default TIMEMASTER in IMAC L is PCU1. For special purposes this can be adjusted to be a different PCU. Two basic settings are necessary for the TIMEMASTER. One is in AvEdit and the other in the vision2000.ini file.

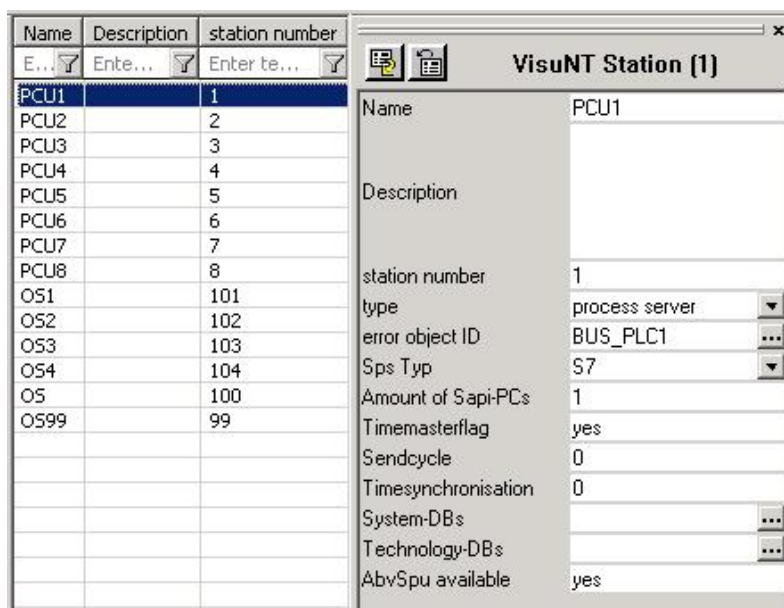
The TimeMaster function is only available for AlphaVision versions from February 2006 or later. If a project is being upgraded from an older version of IMAC L, additional system settings are potentially required (see below in this section).



#### 9.8.8.1. TimeMaster Settings in AvEdit

The AvEdit TIMEMASTER setting can be found in the I55L\_Stations\_List.stn.xml list. For one of the PCUs in the IMAC L system the property 'Timemasterflag' has to be set to 'yes', while all other PCUs must have this property set to 'no':

To edit this Property :

- Double-Click on the 'I55L\_Stations\_List.stn.xml' file.
- The middle window will show all IMAC L STATIONS defined in this project (PCU, OS).



- Click on the PCU to be modified
- Edit the 'Timemasterflag' property by double-clicking on it (toggle between 'yes' and 'no').
- Click on the 'Apply' button () in the right window to save the changes to this station.
- Repeat for all other PCUs to be modified (one PCU = 'yes', all others = 'no')
- Click on the 'Save' button () of AvET.

The PCU which has the 'Timemasterflag' attribute set to 'yes' will be the TIMEMASTER.

### **Important :**

Precisely ONE PCU in an IMAC L system must have its 'Timemasterflag' attribute set to 'yes'. All other PCUs have to have this attribute set to 'no' ! Disregarding this will cause serious malfunction in time setting !

### **9.8.8.2. TimeMaster Settings in VISION2000.INI**

The second setting required for the TIMEMASTER is in the vision2000.ini file. Each OS needs to know, which PCU is the TIMEMASTER. This setting can be found in the vision2000.ini file in section [TS]. The entry 'station' in this section refers to the PCU-number being set to be the TIMEMASTER :

```
;time
[TS]
DateDown=DAD
UtcDown=UTD
LtDown=LTD
DateUp=DAU
UtcUp=UTU
LtUp=LTU
station=1
cycle=3600000
correct=10
```

Edit the 'station' parameter to reflect the new TIMEMASTER settings. Do not modify any of the other settings. Save your settings. Restart each OS for which the vision2000.ini has been modified.

### **9.8.8.3. TimeMaster additional System Settings for Upgrade from old Versions**

Some settings are necessary in the IMAC L system to make the TIMEMASTER function work properly. The newest template files will carry these settings by default. For older projects being updated to a new Alpha-Vision version it may be required to add the settings as follows:

#### **Parameter 'PortDT' in vision2000.ini**

To modify this entry, proceed as follows :

- Click on 'Start → Programs → Accessories → Windows Explorer'
- Navigate to the directory C:\vision\bin
- Double-Click on C:\vision\bin\vision2000.ini  
(an edit window will open, displaying the contents of this file)
- Navigate to the section [Udp2Av] that is located close to the end of the file
- Add the entry 'PortDT' to be as shown below.
- Click on File → Save
- Click on File → Exit
- Reboot the OS

A typical [Udp2Av] section might look like this ('PortDT' entry is highlighted in this document only):

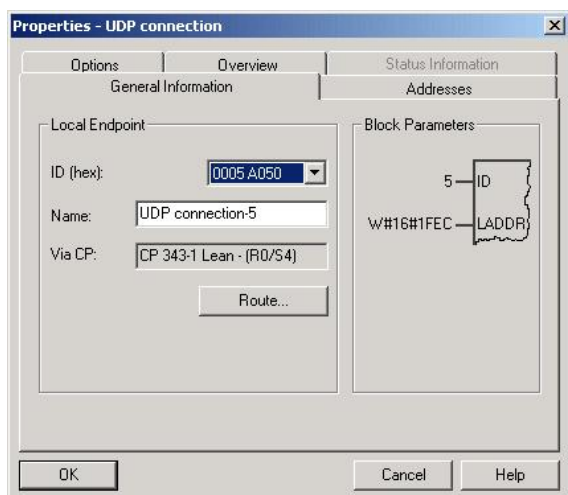
```
[Udp2Av]
PortPlcRead=2000
PortPlcWrite=3000
PortOsRead=2200
PortOsWrite=2200
PortDT=2222
Multicast=239.0.1.0
Buses=1
Stations=1, 2, 3, 101, 102
LifeSignCycle=1500
LifeSignTimeOut=6000
ReadQueueCycle = 50
LogFileSize=10000000
```

### Ethernet Connection for Time Synchronization (in each PLC)

- In the Simatic Manager : Navigate to the root directory of the project
- Double-Click on 'Ethernet(1)' (Simatic NetPro Window opens)
- In the upper left window click on the S7-300 CPU
- The lower left window will show all UDP connections of IMAC L. Check, if this configuration has four or five UDP connections. If the fifth connection named '0005' is missing, it has to be added (instruction see below on this page).
- After completion click on 'Network' → 'Save & Compile'
- Close the NetPro window
- Stop the PLC, Download the System Configuration to the PLC, Start the PLC
- Repeat all steps for all other PLCs

If the UDP-connection #0005 is missing, add this new connection as follows:

- Click on the next empty row below connection '0004 A050'
- Right-Click on 'Insert New Connection'
- In the upper window of the dialogue click on 'All Multicast Stations'
- In the lower window select Type = 'UDP Connection'
- Make sure that the CheckMark 'Display Properties before inserting' is NOT checked
- Click on 'Apply'
- Click on 'OK' in the upcoming warning windows
- Click on 'Close'
- Double-Click on the newly inserted connection '0005'
- Edit the connection properties in the tab 'General Information' to be as follows:



- Edit the connection properties in the tab 'Addresses' to be as follows:



- Click on 'OK' to accept changes
- Save, compile & load the modified hardware configuration

### Check the Settings in DB196

- Edit the following cells in DB196:
  - DB196.DBW32 = 5 (connection number from NetPro)
  - DB196.DBX36.0 = TRUE (PLC =1, cyclical transmission of time telegrams)
  - DB196.DBX36.0 = FALSE (PLC >1, no cyclical transmission of time telegrams)
- Click on 'View' → 'Data View'
- Click on 'Edit' → 'Initialize Data Block'
- Click on 'File' → 'Save'
- Close the DB Editor
- Stop the PLC
- Download DB196 to the PLC
- Start the PLC
- Repeat all steps for all other PLCs

### Check the 'Time Master' setting in DB195

DB 195 is being generated from AvEdit. Normally this should contain the correct settings for the TIMEMASTER flag (see above). Nevertheless older versions of AlphaVision did not generate this correctly. Please check if these settings are correct:

- DB195.DBX165.0 = TRUE for the TimeMaster PLC
- DB195.DBX165.0 = FALSE for all other PLCs

If your version of AlphaVision does generate this wrongly, you will have to adjust this bit for each PCU after generating / compiling the AbvSPU ! Please remember to initialize the DB after modifying this bit.

## 10 Programming Simatic S7-Code for Process Control

### 10.1 Introduction to Process Control Functions

The process control software can manage several kinds of motors and valves. Each motor, valve, standby-unit and tank volume calculation is represented with a specific and unequivocal object in the software. Each object type has its own standard function block (FB80..FB85). These standard function blocks are adjustable in a small range by "option switches" (please refer to the description of these option switches in the respective section of this object type).

While standard function blocks are existing just one time in the software they are multiplexed by several sets of parameter data blocks (each object have its own parameter set).

To save address space in the PLC you will find the object specific parameter data blocks of one object type (motor / valve / standby / tank) lined up in one data block (DB80..DB85, "multi instance data blocks"). You just have to give the standard function block the serial number of the object and it will address the right area in the parameter data block by a pointer.

Each object type has its own function block (FB), which is used to call all objects of this type (FB180..FB185). Each of these FBs has one instance data block assigned to it (DB281..DB285).

The distributed template software includes a basic number of objects of each type for one PCU, which should be sufficient for standard applications:

Object-Type	Number of Predefined Objects in the template distribution
FB80 : IMAC_MOTOR1	50 objects per PCU
FB81 : IMAC_MOTOR2	25 objects per PCU
FB82 : IMAC_MOTOR3	25 objects per PCU
FB83 : IMAC_VALVE	100 objects per PCU
FB84 : IMAC_STANDBY	10 objects per PCU
FB85 : IMAC_TANK	10 objects per PCU

The number of objects can be easily extended [→ 10.1.6 Extending the Number of Objects on page 144].

We recommend to keep the given block names in your software to minimize your engineering work.

#### **Note:**

- 1) The standard function blocks are compiled with the Simatic Managers 'Know-How-Protect' function. All other blocks can be modified and extended by the user.
- 2) The standard function block have to compute the address pointer to the object parameter space. In the basic design of this software there is a dependence between standard function block and parameter data block: FB80 looks at DB80, FB81 looks at DB81 and so on. This basic behavior is implemented in the standard function blocks and cannot be modified by the user.
- 3) The standard process control software is free of standard S7 PLC timer, counter and memory resources (all data is stored in (instance-) data blocks).



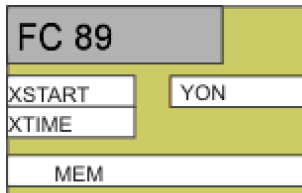
### 10.1.1 Software Block Overview

Object name	Symbolic name	Created	Size in the work-space	Type	Name (Header)
FB80	IMAC_MOTOR1	STL	2426	Function Block	MOTOR1
FB81	IMAC_MOTOR2	STL	3204	Function Block	MOTOR2
FB82	IMAC_MOTOR3	STL	4824	Function Block	MOTOR3
FB83	IMAC_VALVE	STL	3126	Function Block	VALVE
FB84	IMAC_STANDBY	STL	2294	Function Block	STANDBY
FB85	IMAC_TANK	STL	1034	Function Block	TANK
FB90	IMAC_PULSETIMER	SCL	1492	Function Block	P_TIMER
FB180	IMAC_MOTOR1_CALL	STL	8256	Function Block (FB80 calls)	
FB181	IMAC_MOTOR2_CALL	STL	3540	Function Block (FB81 calls)	
FB182	IMAC_MOTOR3_CALL	STL	3540	Function Block (FB82 calls)	
FB183	IMAC_VALVE_CALL	STL	19874	Function Block (FB83 calls)	
FB184	IMAC_STANDBY_CALL	STL	1760	Function Block (FB84 calls)	
FB185	IMAC_TANK_CALL	STL	1800	Function Block (FB85 calls)	
FC89	IMAC_DELAYTIMER	SCL	254	Function	DLYTIMER
DB80	IMAC_MOTOR1_RECORD	DB	5236	Data Block for FB80 calls	
DB81	IMAC_MOTOR2_RECORD	DB	2736	Data Block for FB81 calls	
DB82	IMAC_MOTOR3_RECORD	DB	2636	Data Block for FB82 calls	
DB83	IMAC_VALVE_RECORD	DB	10436	Data Block for FB83 calls	
DB84	IMAC_STANDBY_RECORD	DB	776	Data Block for FB84 calls	
DB85	IMAC_TANK_RECORD	DB	476	Data Block for FB85 calls	
DB90	IMAC_INSTANCE_PULSEFLAGS	DB	104	Instance data block for FB90	
DB280	IMAC_MOTOR1_INSTANCE	DB	6836	Instance data block for FB180	
DB281	IMAC_MOTOR2_INSTANCE	DB	3586	Instance data block for FB181	
DB282	IMAC_MOTOR3_INSTANCE	DB	3536	Instance data block for FB182	
DB283	IMAC_VALVE_INSTANCE	DB	13836	Instance data block for FB183	
DB284	IMAC_STANDBY_INSTANCE	DB	1136	Instance data block for FB184	
DB285	IMAC_TANK_INSTANCE	DB	696	Instance data block for FB185	
DB380	IMAC_MOTOR1_I/O	DB	536	I/O data block for FB80 calls	
DB381	IMAC_MOTOR2_I/O	DB	286	I/O data block for FB81 calls	
DB382	IMAC_MOTOR3_I/O	DB	286	I/O data block for FB82 calls	
DB383	IMAC_VALVE_I/O	DB	1036	I/O data block for FB83 calls	
DB384	IMAC_STANDBY_I/O	DB	176	I/O data block for FB84 calls	
DB385	IMAC_TANK_I/O	DB	356	I/O data block for FB85 calls	

### 10.1.2 Special Function Blocks

#### 10.1.2.1 Delay Timer

The FC89 'IMAC\_DELAYTIMER' is a standard function to replace standard S7 timer. It is used in most of the standard process function blocks (sub routine). However it also can be used for other (user) purposes.



I/O Parameter description:

Parameter	Type	Format	Description
XSTART	IN	BOOL	Activation flag (rising edge)
XTIME	IN	TIME	Delay time in S7 time format
MEM	IN/OUT	STRUC	Memory (structure)
MEM. sdiStartTime	IN/OUT	DINT	Memory: Start time (time for rising edge of XSTART)
MEM. sbEdgeX	IN/OUT	BOOL	Memory: Help flag
YON	OUT	BOOL	Output: 1 if delay time exceeded

Example of function call:

```

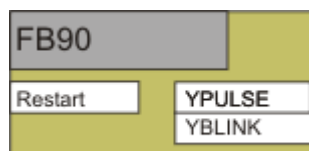
CALL "DELAYTIMER"           //Timer call
XSTART :=M111.0             //Start flag
XTIME  :=T#10S              //Delay time
YON    :=M120.0             //Output
MEM    :=#Memory            //Timer memory
    
```

### 10.1.2.2 Pulse Timer

The FB90 'PULSETIMER' is a pulse generator with up to 8 different pulse lengths (settable via option value). It provides signals with the same pulse/pause time and flashing (one cycle) signals.

#### Note:

- 1) This FB needs an instance data block. In the basic software package the DB90 is the instance data block for the FB90. It is strictly recommended to call the FB90 precisely once with the DB90 in the cycle (please do not change time values in the DB90).
- 2) The outputs of FB90 are free for common purposes. You can address the outputs directly to the DB90 workspace or by a connected memory field to the output of the FB90.
- 3) It is possible to call several instances (with different instance DB's) from FB90.
- 4) The cycle time should be minimum 0,5 x shortest adjusted pulse time .



I/O Parameter description:

Parameter	Type	Format	Description
Restart	IN	BOOL	Restart flag to reset all timer memories (rising edge)
YPULSE	OUT	BYTE	Output with 8 flashing flags
YBLINK	OUT	BYTE	Output with 8 alternating flags

Instance data block:

DB90 -- TechnFct_S7\Generator_CPU315\CPU 316-2 DP						
	Address	Declaration	Name	Type	Initial value	Comment
1	0.0	in	Restart	BOOL	FALSE	Restart timer
2	2.0	out	YPulse	BYTE	B#16#0	Pulse output (one cycle)
3	3.0	out	YBlink	BYTE	B#16#0	Blink output (equal on/off)
4	4.0	stat	POPT	STRUCT		
5	+0.0	stat	Timer00	TIME	T#500MS	
6	+4.0	stat	Timer01	TIME	T#1S	
7	+8.0	stat	Timer02	TIME	T#2S	
8	+12.0	stat	Timer03	TIME	T#5S	
9	+16.0	stat	Timer04	TIME	T#10S	
10	+20.0	stat	Timer05	TIME	T#20S	
11	+24.0	stat	Timer06	TIME	T#1M	
12	+28.0	stat	Timer07	TIME	T#1H	
13	=32.0	stat		END_STRUCT		
14	36.0	stat	sdiStartTime	ARRAY[0 .. 7]		Start time store
15	*4.0	stat		DINT		Start time store

Option field inside instance data block:

Parameter	Type	Format	Description
POPT.Timer00	IN	TIME	Pulse time value timer 0
:	:	:	
POPT.Timer07	IN	TIME	Pulse time value timer 1

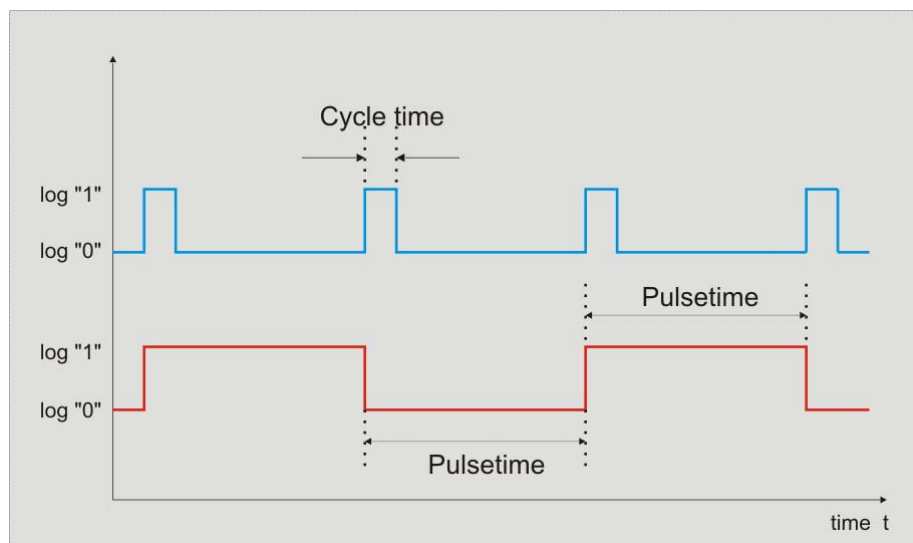


Fig.: Output diagram Pulse timer

Bit 0	Timer00
Bit 1	Timer01
Bit 2	Timer02
Bit 3	Timer03
Bit 4	Timer04
Bit 5	Timer05
Bit 6	Timer06
Bit 7	Timer07

Fig.: Output YBLINK

Bit 0	Timer00
Bit 1	Timer01
Bit 2	Timer02
Bit 3	Timer03
Bit 4	Timer04
Bit 5	Timer05
Bit 6	Timer06
Bit 7	Timer07

Fig.: Output YPULSE

**Note to Fig. "Output diagram Pulse timer":**

**Blue graph:** Flashing pulses (output is "1" for one cycle after running out pulse time)

**Red graph:** Alternating pulse (output changes state after running out pulse time)

Example of function call (as implemented in the Template distribution in OB1):

```
CALL "PULSETIMER", "Instanz Pulse flags"
Restart  :=          //Restart flag
YPulse   := DB89.DBB0 //Flashing flags
YBlink   := DB89.DBB1 //Alternating / blinking flags
```

The bits in DB89 / DBB 0 + 1 can be used by any program (e.g. to produce a blinking lamp) by simply reading the required bit.

## 10.1.3 Software Structure

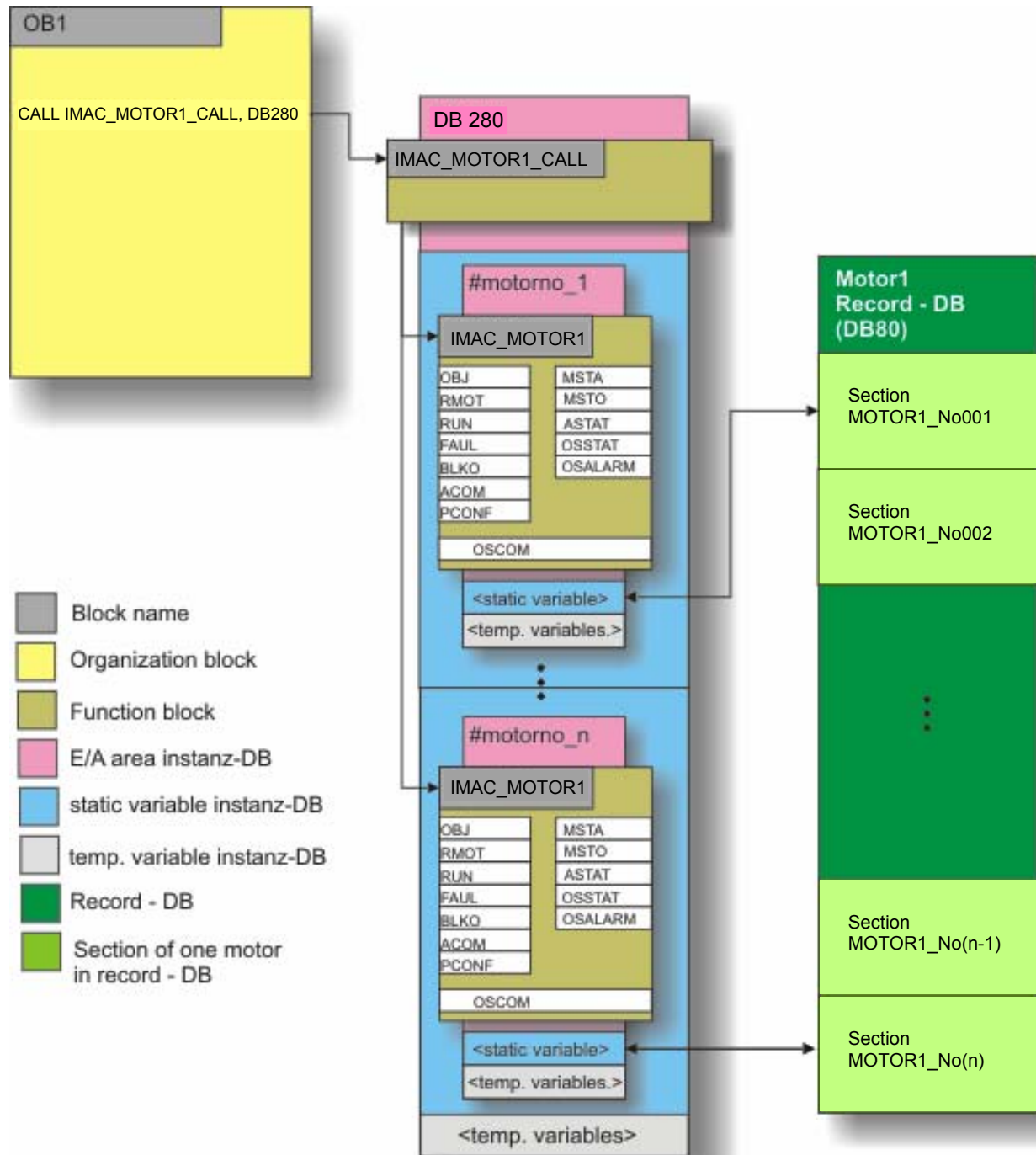


Figure: Typical software structure of process control software

From the OB1 the object type distributor FB180 (FB181...FB185) is called. The FB180 (FB181...FB185) uses a multi- instance data block. Inside the FB180 standard function blocks (e.g. FB80 = MOTOR1) are called. Each standard function call uses one section of the multi- instance- data block and exchanges parameter data with one section of the respective record data block.

### 10.1.4 Memory Distribution

For standard process objects you have two types of memories to manage:

1) I/O Data

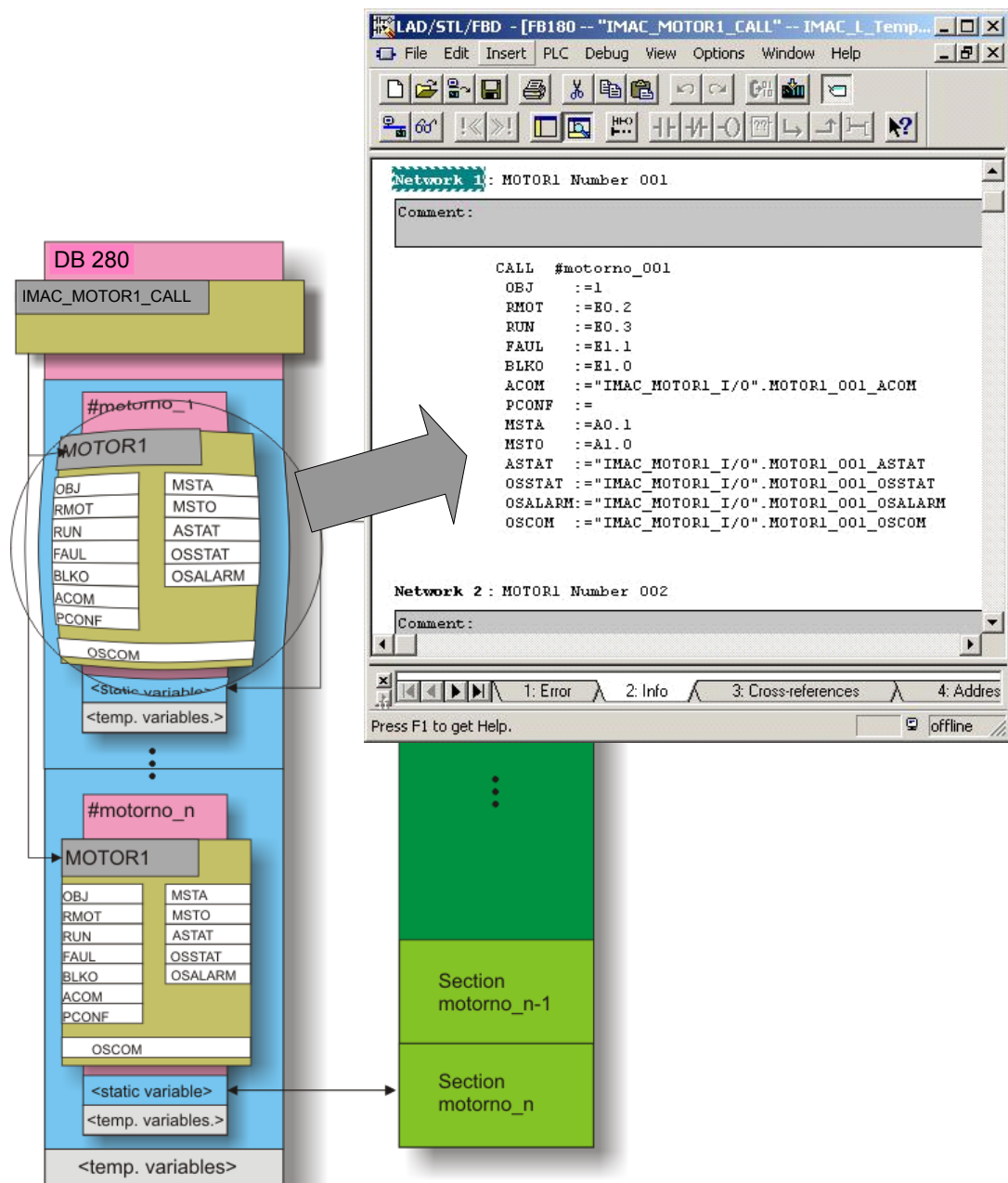


Figure: I/O data for standard process function blocks

Most of the I/O data is variable to the run time:

- Process values from and to the process
- Data words from and to the OS (these data are allocated in the I/O Data Blocks DB38x)

The I/O data (inputs and outputs) have to be connected in the FB180 ... FB185 by directly entering them to the respective function call. For all predefined objects in the template distribution, the connection to the OS (DB38x) is already contained in the function call.



## 2) Parameter Data (record data)

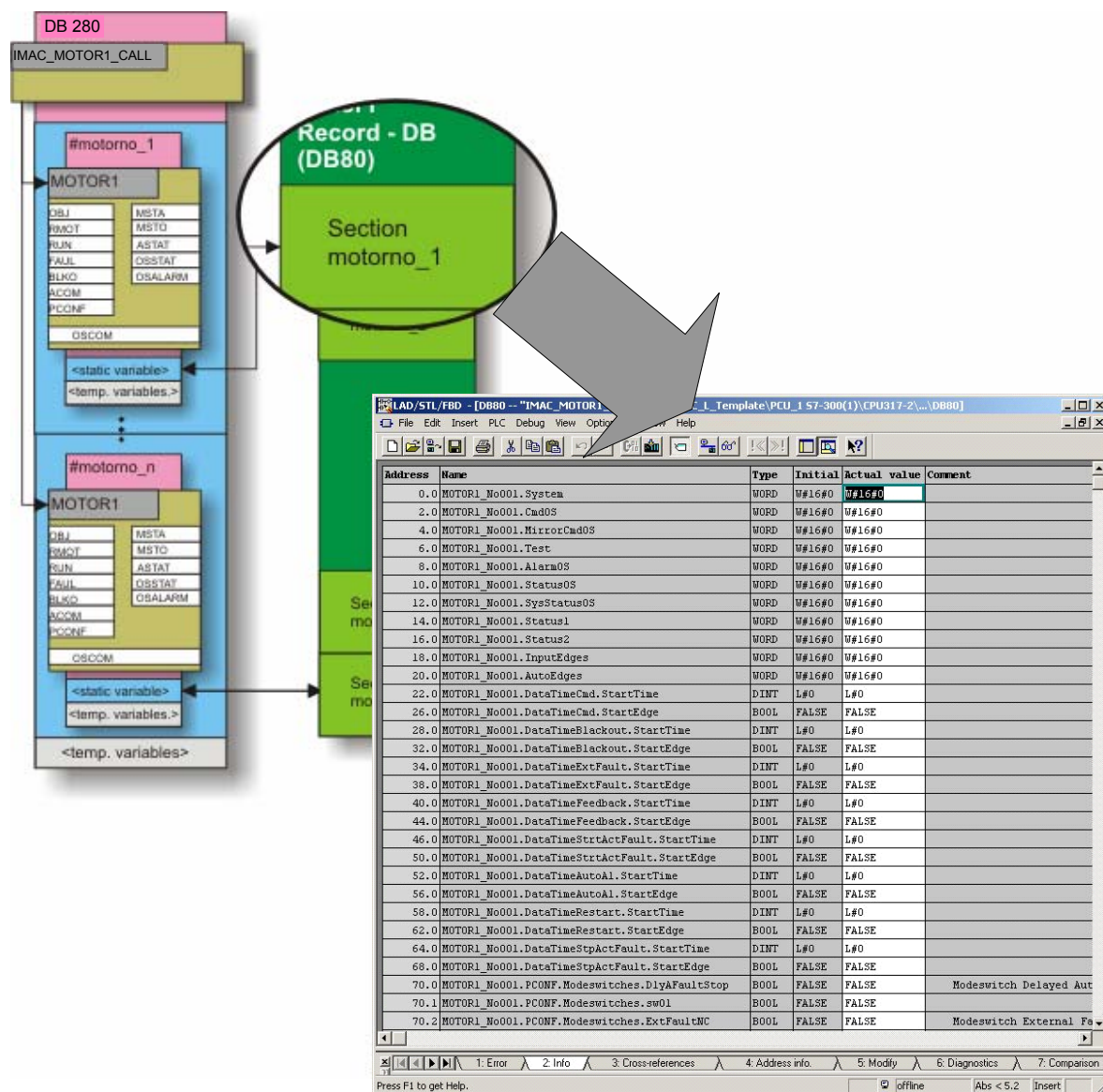


Figure: Parameter data in the record data block

All static data of the process control function blocks are stored in the respective record data blocks (DB80...DB85). There are two sections of data for each function block instance :

1. All those data stored in the structures **PCONF** (named PCONF.xyz) have to be parameterized by the user (refer to the description later in the object typesw descriptions).
2. All other data in these record data blocks (i.e. outside of the PCONF structures) are static data used by the function blocks for internal purposes.

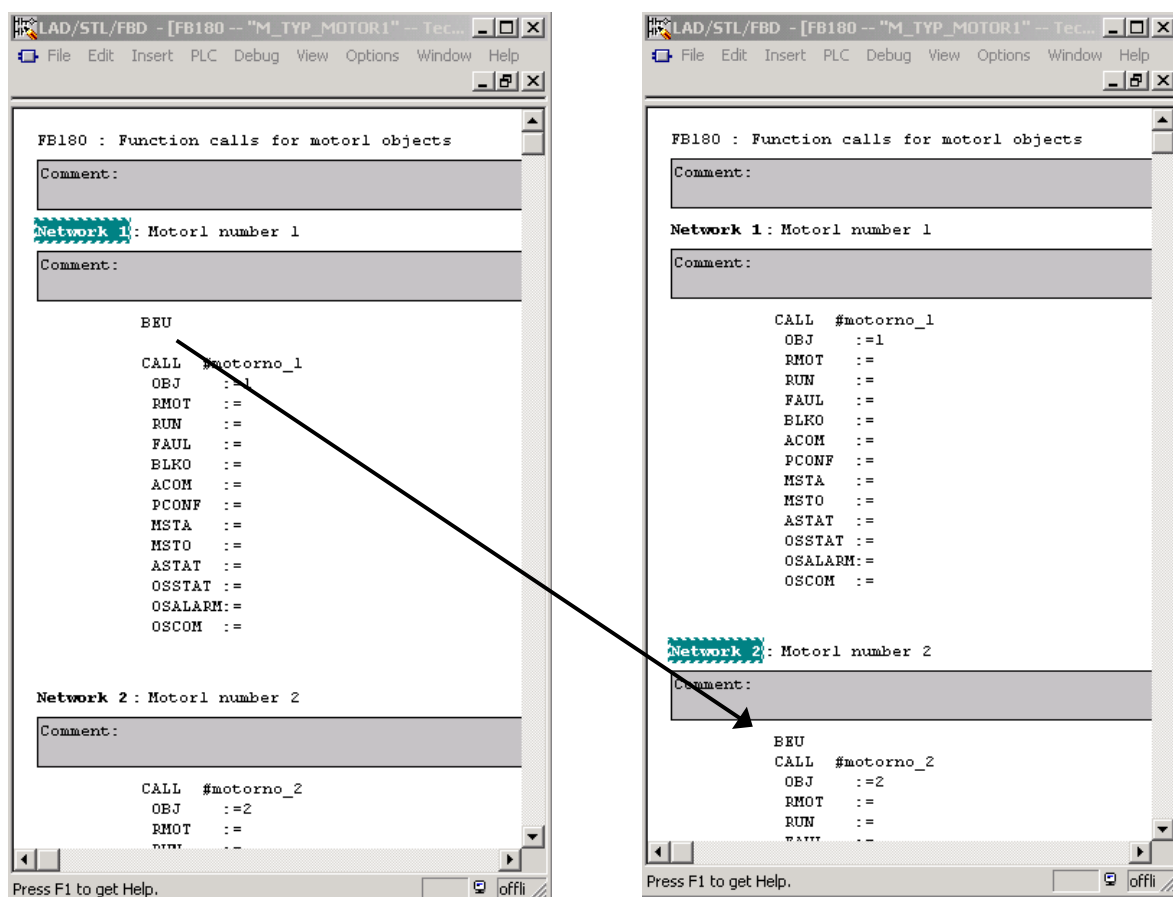
⚠ **Do not change any data outside of the PCONF structures !**

### 10.1.5 Activation of Template Objects

The basic software is distributed with a number of predefined objects per object type. All except for some example objects are deactivated in the original template distribution.

This chapter will describe how to activate one object (example: Motor1):

- Open S7 Manager with your project data and standard process control software
- Open FB180 (object call of all Motor1 objects)
- Move "BEU" command from current (last active) network to the following network.



- Parameterize (customize) I/O parameter of function block
- Parameterize option data in record DB (in this case DB80). Take care about the right object type (DB number), object number (offset in DB) and parameter configuration area (parameter are beginning with PCONF.xyz). Please refer to the next figure!

**Note:** If you are changing parameters in a running system you should do your change online (record DB stored in the PLC). This is the only possibility to keep all static variables (e.g. timer values) after the download in the record data block. Update parallel also the offline data block.

Object type

Running object number of object type

User configuration area

Object type	Running object number of object type	User configuration area	Value 1	Value 2	Value 3
MOTOR1_No1	44.0	DateTimeFeedback.StartEdge	BOOL	FALSE	FALSE
MOTOR1_No1	46.0	DateTimeStrtActFault.StartTime	DINT	L#0	L#0
MOTOR1_No1	50.0	DateTimeStrtActFault.StartEdge	BOOL	FALSE	FALSE
MOTOR1_No1	52.0	DateTimeAutoAl.StartTime	DINT	L#0	L#0
MOTOR1_No1	56.0	DateTimeAutoAl.StartEdge	BOOL	FALSE	FALSE
MOTOR1_No1	58.0	DateTimeRestart.StartTime	DINT	L#0	L#0
MOTOR1_No1	62.0	DateTimeRestart.StartEdge	BOOL	FALSE	FALSE
MOTOR1_No1	64.0	DateTimeStpActFault.StartTime	DINT	L#0	L#0
MOTOR1_No1	68.0	DateTimeStpActFault.StartEdge	BOOL	FALSE	FALSE
MOTOR1_No1	70.0	PCNF.Modeswitches.DlyAFaultStop	BOOL	FALSE	FALSE
MOTOR1_No1	70.1	PCNF.Modeswitches.sw01	BOOL	FALSE	FALSE
MOTOR1_No1	70.2	PCNF.Modeswitches.ExtFaultNC	BOOL	FALSE	FALSE
MOTOR1_No1	70.3	PCNF.Modeswitches.sw03	BOOL	FALSE	FALSE
MOTOR1_No1	70.4	PCNF.Modeswitches.sw04	BOOL	FALSE	FALSE
MOTOR1_No1	70.5	PCNF.Modeswitches.sw05	BOOL	FALSE	FALSE
MOTOR1_No1	70.6	PCNF.Modeswitches.sw06	BOOL	FALSE	FALSE
MOTOR1_No1	70.7	PCNF.Modeswitches.sw07	BOOL	FALSE	FALSE
MOTOR1_No1	71.0	PCNF.Modeswitches.NoConstOutput	BOOL	FALSE	FALSE

- f) Download record data block (if you did not change it online)
- g) Download FB180 (object call of all Motor1 objects)
- h) Check functionality

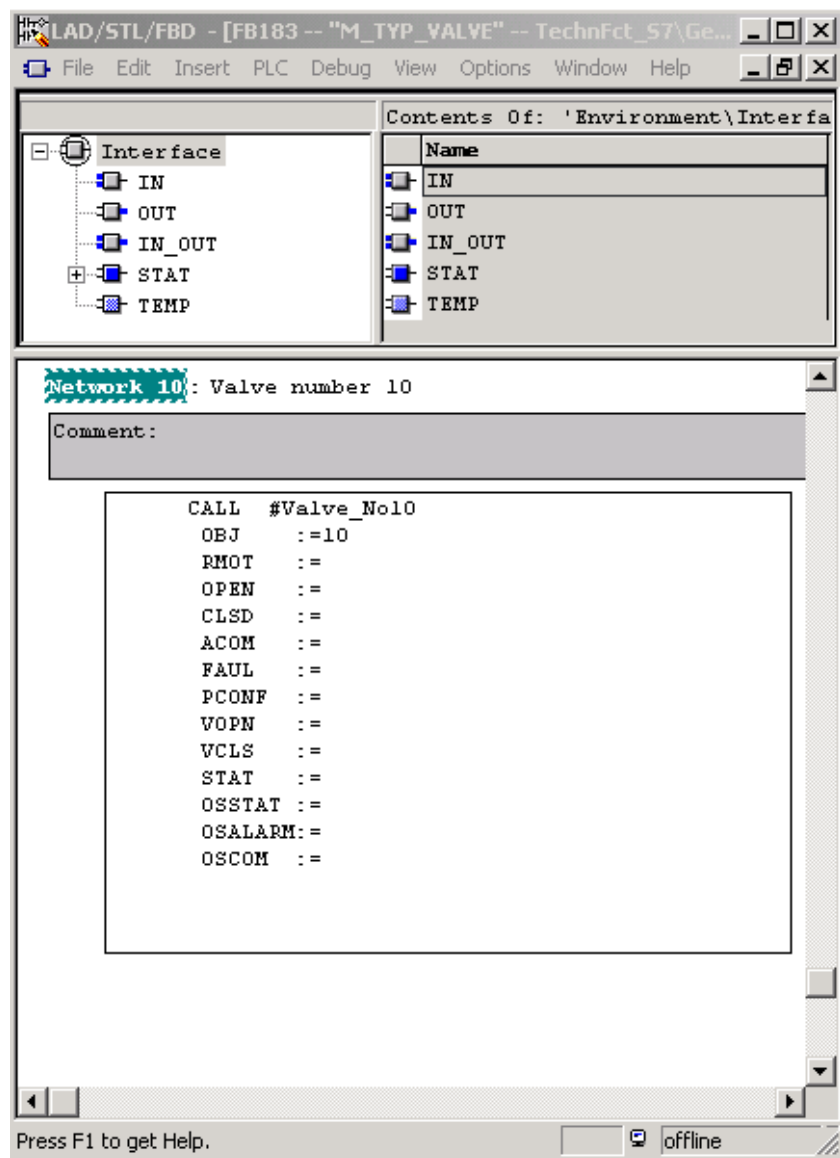
### 10.1.6 Extending the Number of Objects

If more than the predefined number of objects for one process control type are required in one PCU, some blocks in the process control software have to be extended.

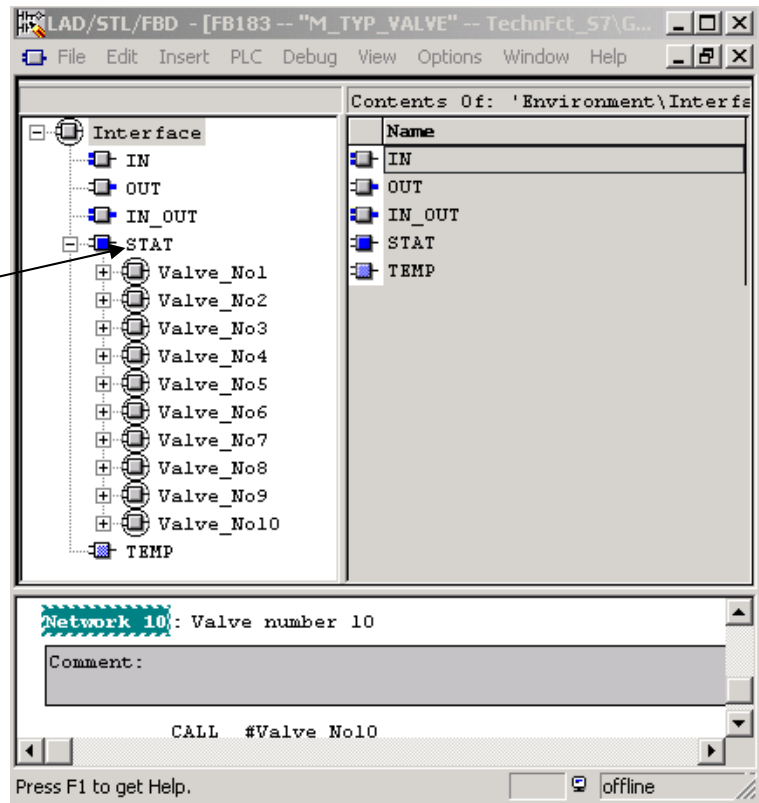
**Note:** This action should only be executed in a safe condition of your plant. After download the new offline data will overwrite some static variables (like timer memories, status information's and edge detections). This may cause undefined actions of your process system!

Please follow these instructions step by step (as an example we create valve object number 11, where only 10 valves were existing before. The screenshots in this example are not completely identical with the control software version distributed with the template files):

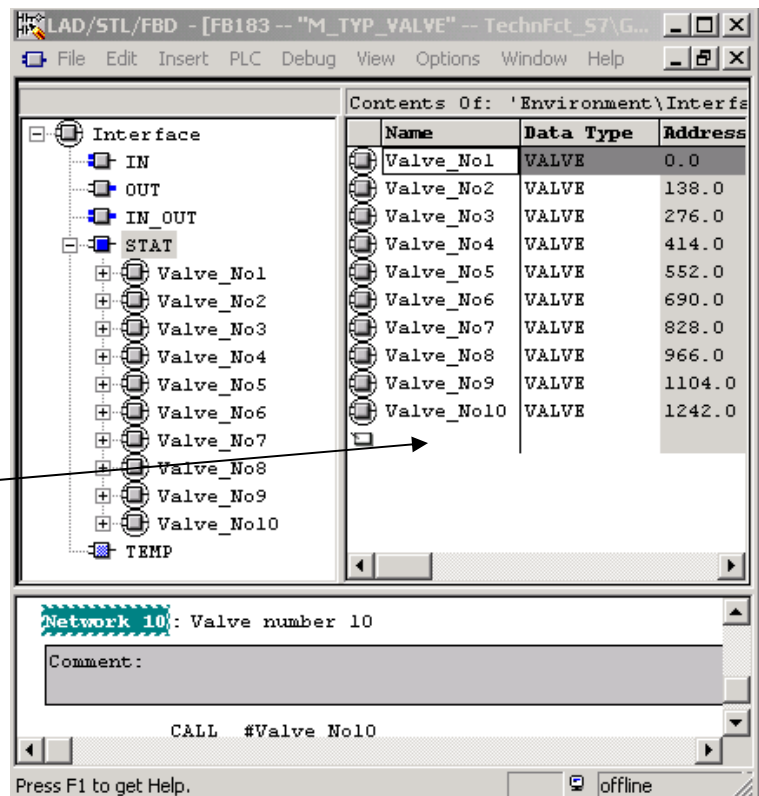
- Open S7 Manager with your project data and standard process control software
- Open FB183 (object call of all valve (flaps) objects)



- e) Click on text „STAT“



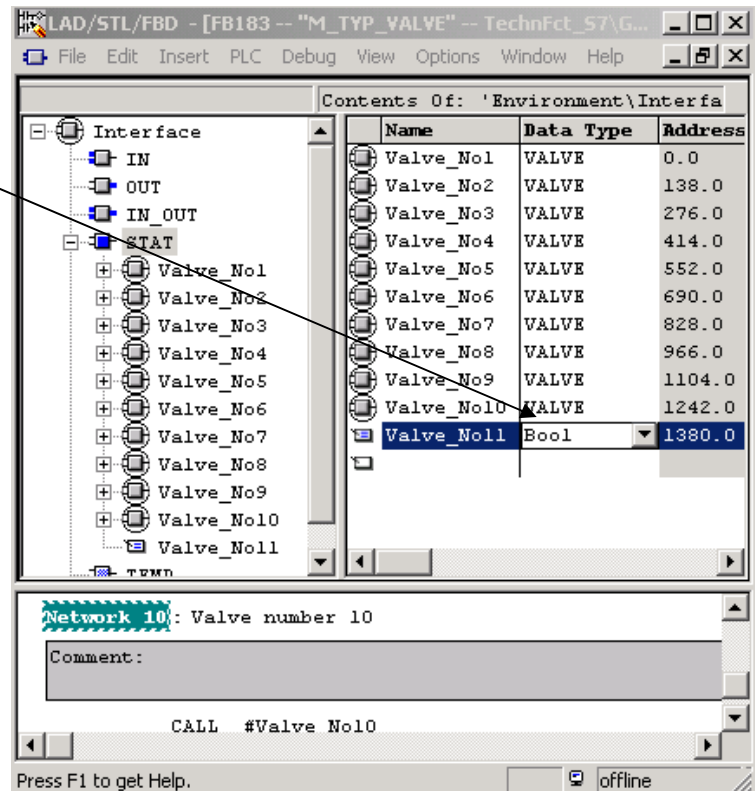
- f) Click on empty field below last object to add a new object name.  
Type „Valve\_No11“ followed by a <TAB> key





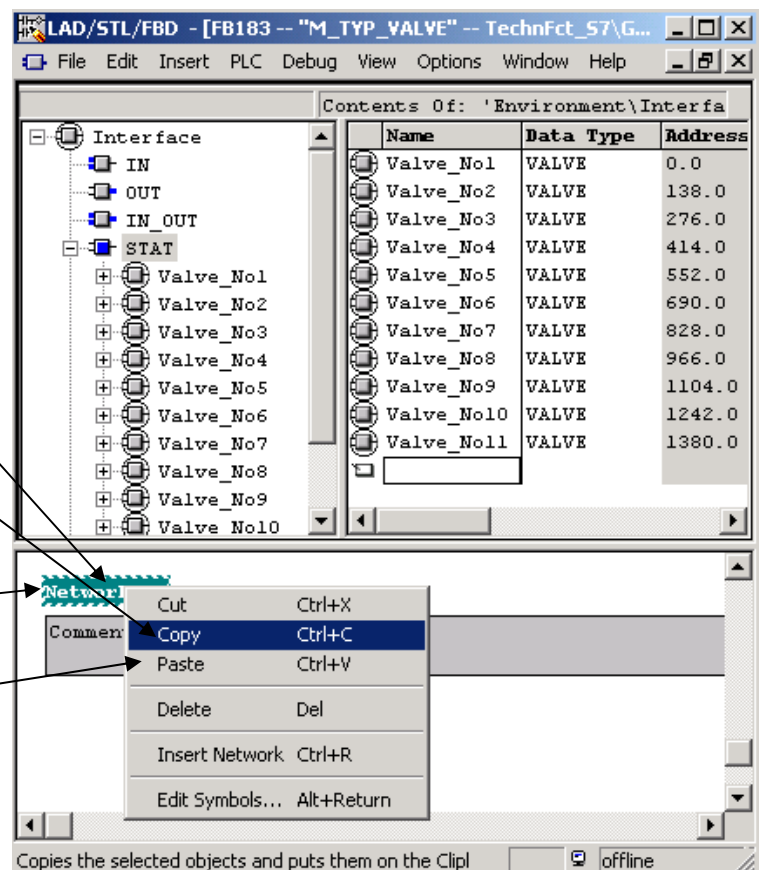
- g) Click in the „Data Type“ column on the text „BOOL“ and enter the standard object function block number of this object type (in this case: FB83).  
Close this action with <ENTER>

**Note:** You also can enter the symbol „VALVE“



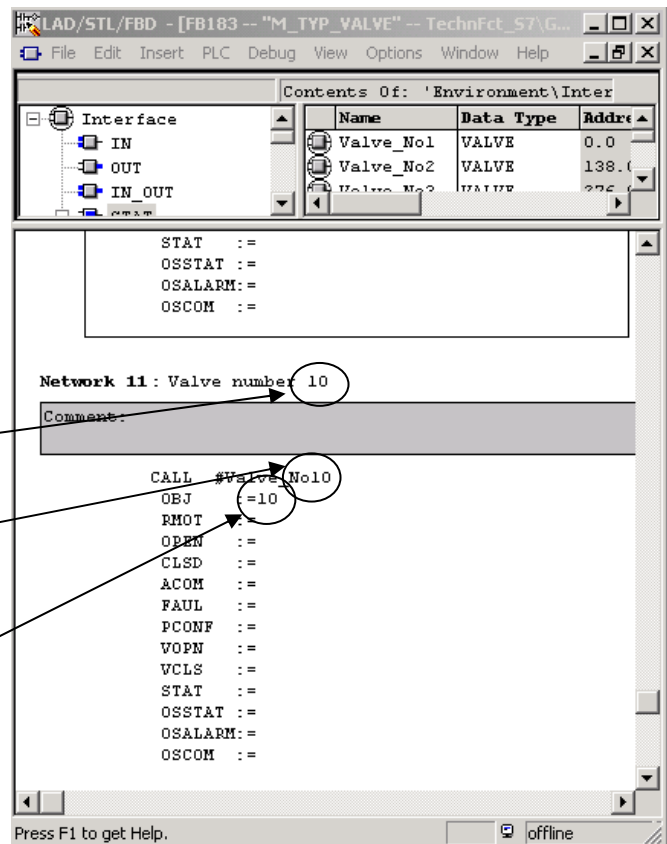
- h) Click with the right mouse button on the last network head and select „Copy“.

- i) Click with the right mouse button on the last network head and select „Paste“.



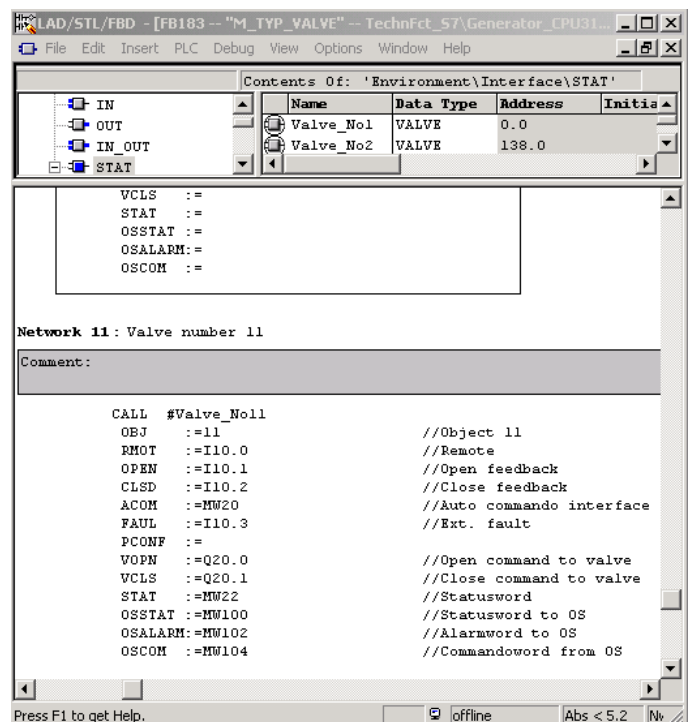


- j) Change  
network description  
object name  
running object number



- k) Enter also your I/O variable  
and save and close your file.

Fig. beside shows an  
example of a complete func-  
tion block call of a valve.



- l) Open DB83 (record data block of all valve (flaps) objects) in the "Declaration View"

**Note:** If the Simatic Manager opens this DB in the "Data View": you can change to the "Declaration View" by pressing <Ctrl> <5>

- m) A click on addressfield „+0.0“ will select valve record number 1.

Press right mouse button on the same field to get context menu. Select „Copy“

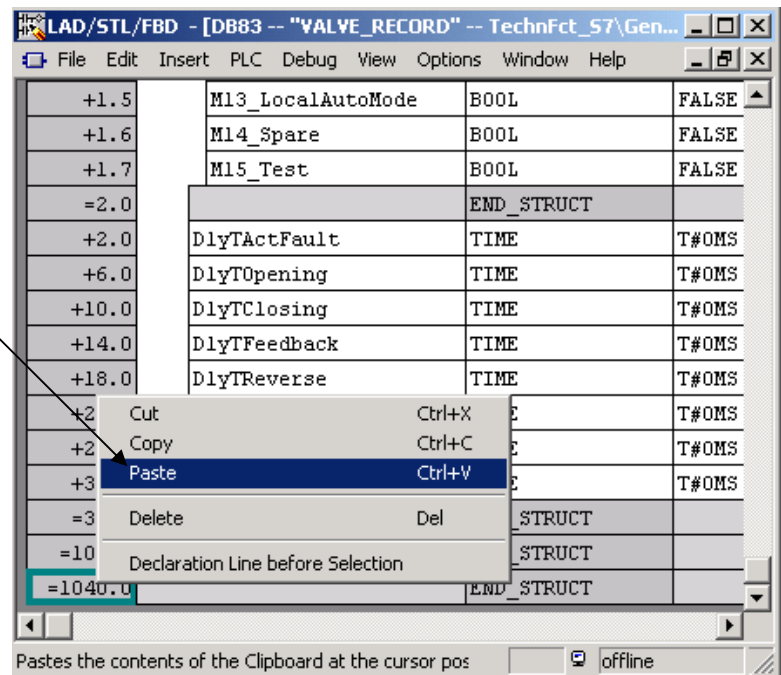
Address	Name	Type	Initial	Comment
0.0		STRUCT		
+0.0	VALVE_No1	STRUCT		
+0.0	System	WORD	W#16#0	
+2.0	CmdOS	WORD	W#16#0	
+4.0	MirrorCmdOS	WORD	W#16#0	
+6.0	Test	WORD	W#16#0	
+8.0	AlarmOS	WORD	W#16#0	
+10.0	StatusOS	WORD	W#16#0	
+12.0	SysStatusOS	WORD	W#16#0	
+14.0	Status1	WORD	W#16#0	
+16.0	Status2	WORD	W#16#0	
+18.0	Status3	BYTE	B#16#0	
+19.0	InputEdges	BYTE	B#16#0	
+20.0	AutoEdges	WORD	W#16#0	

- n) Move down to the end of data block (slider)

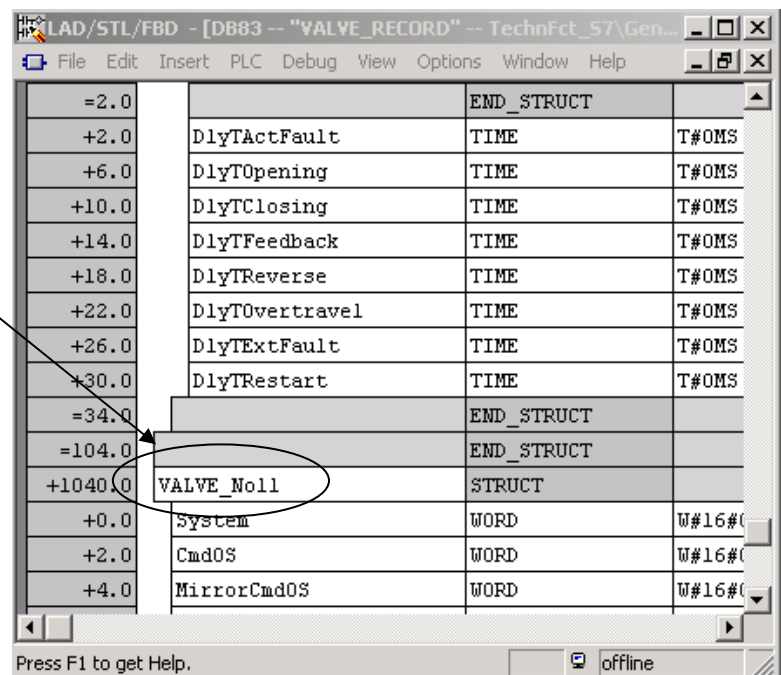
Select with the left mouse button the last address field. Then press the right mouse button.

+1.5	M13_LocalAuto	BOOL	FALSE	Set automati
+1.6	M14_Spare	BOOL	FALSE	Spare
+1.7	M15_Test	BOOL	FALSE	Testbit (1)
=2.0		END_STRUCT		
+2.0	DlyTActFault	TIME	T#0MS	Setpoint Act
+6.0	DlyTOpening	TIME	T#0MS	Setpoint Wa
+10.0	DlyTClosing	TIME	T#0MS	Setpoint Wa
+14.0	DlyTFeedback	TIME	T#0MS	Setpoint fee
+18.0	DlyTReverse	TIME	T#0MS	Setpoint ch
+22.0	DlyTOvertravel	TIME	T#0MS	Setpoint aut
+26.0	DlyTextFault	TIME	T#0MS	Setpoint ext
+30.0	DlyTRestart	TIME	T#0MS	Setpoint res
=34.0		END_STRUCT		
=104.0		END_STRUCT		
+1040.0		END_STRUCT		

- o) Select „Paste“.  
(a new record section  
will be copied to the  
end of the record DB)



- p) Enter a new section  
name and save your  
file.



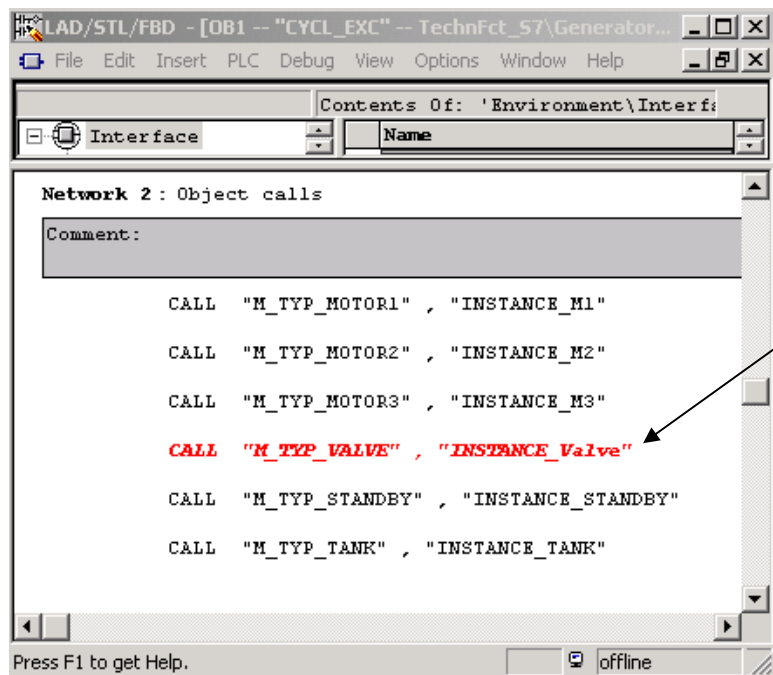
- q) Change to the „Data View“ by pressing <Ctrl> <4> and move to the end of the data block

LAD/STL/FBD - [DB83 -- "VALVE\_RECORD" -- TechnFct\_S7\Generator\_CPU315\CPU 316-2 DP\...\DB83]

1098.0	VALVE_No11.DataTimeExtFault.StartTime	DINT	L#0	L#0	
1102.0	VALVE_No11.DataTimeExtFault.StartEdge	BOOL	FALSE	FALSE	
1104.0	VALVE_No11.DataTimeRestart.StartTime	DINT	L#0	L#0	
1108.0	VALVE_No11.DataTimeRestart.StartEdge	BOOL	FALSE	FALSE	
1110.0	VALVE_No11.PCONF.Modeswitches.M0_SingleActingValve	BOOL	FALSE	FALSE	Single acting ve
1110.1	VALVE_No11.PCONF.Modeswitches.M1_ResetMode	BOOL	FALSE	FALSE	Open/Close and F
1110.2	VALVE_No11.PCONF.Modeswitches.M2_OperationMode	BOOL	TRUE	FALSE	AUTO/MANUAL/LOCA
1110.3	VALVE_No11.PCONF.Modeswitches.M3_Status	BOOL	FALSE	FALSE	Change status OF
1110.4	VALVE_No11.PCONF.Modeswitches.M4_LockInput	BOOL	FALSE	FALSE	Commands and fai
1110.5	VALVE_No11.PCONF.Modeswitches.M5_ESO	BOOL	FALSE	FALSE	Open valve if ES
1110.6	VALVE_No11.PCONF.Modeswitches.M6_ReverseTime	BOOL	FALSE	FALSE	Double acting ve
1110.7	VALVE_No11.PCONF.Modeswitches.M7_OvertravelTime	BOOL	FALSE	FALSE	Double acting ve
1111.0	VALVE_No11.PCONF.Modeswitches.M8_Spare	BOOL	FALSE	FALSE	Spare
1111.1	VALVE_No11.PCONF.Modeswitches.M9_OpenAfterBlackout	BOOL	FALSE	FALSE	Open after black
1111.2	VALVE_No11.PCONF.Modeswitches.M10_ExtFaultNC	BOOL	FALSE	FALSE	External Fault i
1111.3	VALVE_No11.PCONF.Modeswitches.M11_Spare	BOOL	FALSE	FALSE	Spare
1111.4	VALVE_No11.PCONF.Modeswitches.M12_Spare	BOOL	FALSE	FALSE	Spare
1111.5	VALVE_No11.PCONF.Modeswitches.M13_LocalAutoMode	BOOL	FALSE	FALSE	Set automatic st
1111.6	VALVE_No11.PCONF.Modeswitches.M14_Spare	BOOL	FALSE	FALSE	Spare
1111.7	VALVE_No11.PCONF.Modeswitches.M15_Test	BOOL	FALSE	FALSE	Testbit (1)
1112.0	VALVE_No11.PCONF.DlyTActFault	TIME	T#0MS	T#0MS	Setpoint Action
1116.0	VALVE_No11.PCONF.DlyTOpening	TIME	T#0MS	T#0MS	Setpoint Waiting
1120.0	VALVE_No11.PCONF.DlyTClosing	TIME	T#0MS	T#0MS	Setpoint Waiting
1124.0	VALVE_No11.PCONF.DlyTFeedback	TIME	T#0MS	T#0MS	Setpoint feedbac
1128.0	VALVE_No11.PCONF.DlyTReverse	TIME	T#0MS	T#0MS	Setpoint change
1132.0	VALVE_No11.PCONF.DlyTOvertravel	TIME	T#0MS	T#0MS	Setpoint auto al
1136.0	VALVE_No11.PCONF.DlyTExtFault	TIME	T#0MS	T#0MS	Setpoint externa
1140.0	VALVE_No11.PCONF.DlyTRestart	TIME	T#0MS	T#0MS	Setpoint restart

Press F1 to get Help. offline Abs < 5.2 Insert

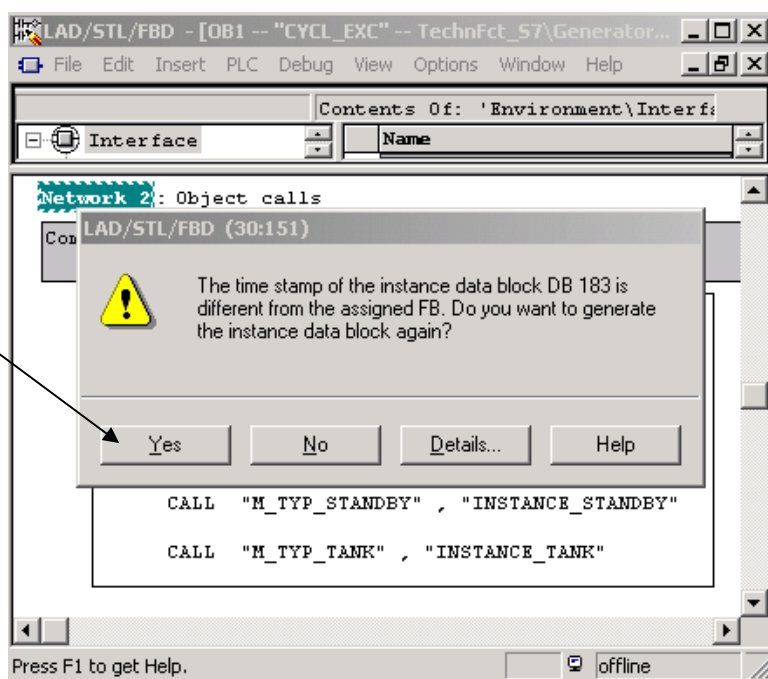
- r) Customize the PCONF (parameter configuration) area of the new record section and close your file (with saving).



s) Open OB1 (cycle distributor) and set your cursor to the end of the red line.

Press the <ENTER> button of your keyboard.

t) Confirm the warning with „Yes“ and save your file.



- u) Extend DB383 (I/O data) to contain one more record (Copy, paste & modify last existing record)
- v) Download your changed files:
  - 1) DB283 (Instance DB of FB183, new generated)
  - 2) DB383 (I/O data to OS)
  - 3) DB83 (Record DB, one section added)
  - 3) FB183 (Object function block distributor of valve objects, one instance and one network added)
  - 4) OB1 (Cycle distributor, time stamp conflict removed)

## 10.2 Valves / Flaps

### 10.2.1 Used Resources

Resource	Description
FB83	Standard object function block
DB83	Record data block (includes all records of type "Valve")
FB183	Distribution function block for objects of type "Valve"
DB283	Instance data block to FB183
DB383	I/O Data to OS
FC89	Delay timer (subroutine for FB83)
SFC20	Block move /subroutine for FB83), block implemented in firmware of PLC

Used memory space [in byte]:

Resource	Local data	MC7	Load memory	Work memory
FB83	74	3090	3826	3126
DB83 (100 objects)		10400	24182	10436
DB83 (one add. object)		+104	+242	+104
FB183 (100 objects)	8	19838	40322	19874
FB183 (one add. object)	+0	min. +116	min. +318	min. +116
DB283 (100 objects)		13800	34080	13836
DB283 (one add. object)		+138	+340	+138
DB383 (100 objects)		1000	2084	1036
DB383 (one add. object)		+10	+20	+10
FC89	12	218	334	254
SFC20	-	-	-	-

**Hint:** Timers, counters and global memory are not used by the process control software.

### 10.2.2 Description I/O Area

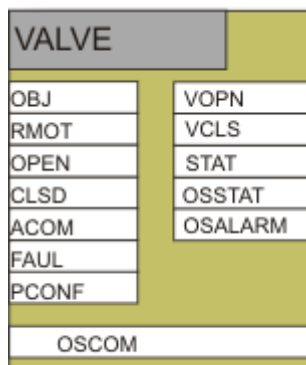


Figure: Block view of valve object with I/O interface

Name	Type	Format	Initial value
<b>OBJ</b>	<b>IN</b>	<b>INT</b>	<b>0</b>
<b>Description</b>			
Serial number of valve object in record DB. Used to point to the beginning of object record data set. Object Number Range is '1...n' for 'n' objects.			



Name	Type	Format	Initial value
<b>RMOT</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Remote' feedback from local panel of valve (process value). 0 -> LOCAL 1 -> REMOTE Valve will only be indicated in local mode (no feedback control, no control access from OS). In remote mode the operator has the possibility to open/close the valve.			

Name	Type	Format	Initial value
<b>OPEN</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Open' feedback from valve (process value). 0 -> Valve not open 1 -> Valve is open			

Name	Type	Format	Initial value
<b>CLSD</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Closed' feedback from valve (process value). 0 -> Valve not closed 1 -> Valve is closed			

Name	Type	Format	Initial value
ACOM	IN	WORD	W#16#0
Description			
Auto commando interface. This interface is only active in REMOTE and AUTO mode (this is an PLC internal interface)			
Bit	Value	Function	Description
0	1		
1	2		
2	4		
3	8		
4	16		
5	32		
6	64		
7	128		
8	256	Open	Command: OPEN to valve
9	512	Close	Command: CLOSE to valve
10	1024		
11	2048	Lock	Command: Block OPEN/CLOSED commands from operator
12	4096	Emergency off	Command: Emergency CLOSE
13	8192	Blackout start	Command: Restart after blackout
14	16384	Pleiger 24V valve	Special functionality for Pleiger 24V valves
15	32768		

Name	Type	Format	Initial value
<b>FAUL</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
External fault input (process signal). Please refer to the mode switch "M10_ExtFaultNC" for the function of this input:			
<u>M10_ExtFaultNC = false:</u>			
0 -> no fault			
1 -> fault			
<u>M10_ExtFaultNC = true:</u>			
0 -> fault			
1 -> no fault			

Name	Type	Format	Initial value
<b>PCONF</b>	<b>IN</b>	<b>STRUC</b>	<b>0</b>
<b>Description</b>			
Record pointer. Do not change this input without consulting Siemens!			

Name	Type	Format	Initial value
<b>VOPN</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Open' command output to valve (process signal). Please refer to mode switch "M0_SingleActingValve" for the function of this output:			
<u>M0_SingleActingValve = false (double acting valve):</u>			
0 -> No command to valve			
1 -> Open command to valve			
<u>M0_SingleActingValve = true (single acting valve):</u>			
0 -> Close command to valve			
1 -> Open command to valve			

Name	Type	Format	Initial value
<b>VCLS</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Close' command output to valve (process signal). Please refer to mode switch "M0_SingleActingValve" for the function of this output:			
<u>M0_SingleActingValve = false (double acting valve):</u>			
0 -> No command to valve			
1 -> Close command to valve			
<u>M0_SingleActingValve = true (single acting valve):</u>			
Output not active			

Name		Type	Format	Initial value
<b>STAT</b>		<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description				
Auto status interface, also active in non automatic states (this is an PLC internal interface)				
Bit	Function	Description		
0				
1				
2				
3				
4				
5				
6				
7				
8	Opened	Valve status: Opened		
9	Closed	Valve status: Closed		
10	Local	Valve status: Local		
11	Manual	Valve status: Manual		
12	Auto	Valve status: Auto		
13	Opening	Valve status: Opening		
14	Closing	Valve status: Closing		
15	Alarm	Valve status: Alarm		

Name		Type	Format	Initial value
<b>OSSTAT</b>		<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description				
Status word to OS icon (internal interface). Connect this output to the interface of your OS.				
Value	Status			
0	Undefined			
1	Opened			
2	Closed			
3	Opening			
4	Closing			
5	Undefined position			
+256	LOCAL Mode (Offset Value)			
+512	MANUAL Mode (Offset Value)			
+1024	AUTO Mode (Offset Value)			

Name	Type	Format	Initial value
<b>OSALARM</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Alarm word to OS icon (internal interface). Connect this output to the interface of your OS.			
Value	Status		
0	Undefined		
1	Normal		
2	External Fault Alarm		
3	Moving Alarm		
4	Opening Alarm		
5	Closing Alarm		
6	Non -Feedback Alarm		
7	Lock		
8	Emergency OFF		

Name	Type	Format	Initial value
<b>OSCOM</b>	<b>IN/OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Command word from OS icon (internal interface). Connect this output to the interface of your OS.			
Bit	Value	Function	Description
0	1	Open	Open command to Valve
1	2	Close	Close command to Valve
2	4	Toggle	Toggle command to Valve
3	8		
4	16		
5	32		
6	64		
7	128	Reset	Reset command to Valve
8	256		
9	512	MANUAL-Mode	MANUAL-Mode Command to Valve
10	1024	AUTO-Mode	AUTO-Mode Command to Valve
11	2048		
12	4096		
13	8192		
14	16384		
15	32768		

### 10.2.3 Description Static Variables (options)

a) Mode switches (to adjust in PCONF in DB83 in structure 'PCONF' for each object)

Mode switch	Format	Preset	Description
M0_SingleActingValve	BOOL	FALSE	Single acting valve (1); Double acting valve (0)
M1_ResetMode	BOOL	FALSE	Open/Close and RESET (1); only RESET (0)
M2_OperationMode	BOOL	FALSE	AUTO/MANUAL/LOCAL(1); Only MANUAL/LOCAL
M3_Status	BOOL	FALSE	Change status OPENED/CLOSED by switching from REMOTE to LOCAL (1)
M4_LockInput	BOOL	FALSE	Commands and failure (1); only failure
M5_ESO	BOOL	FALSE	Open valve if ESO=1 (1); Close valve if ESO=1 (0)
M6_ReverseTime	BOOL	FALSE	Double acting valves: Reverse Time ON
M7_OvertravelTime	BOOL	FALSE	Double acting valves: Over travel Time ON
M8_Spare	BOOL	FALSE	Spare
M9_OpenAfterBlackout	BOOL	FALSE	Open after blackout (1)
M10_ExtFaultNC	BOOL	FALSE	External Fault is NC (1); NO (0)
M11_Spare	BOOL	FALSE	Spare
M12_Spare	BOOL	FALSE	Spare
M13_LocalAutoMode	BOOL	FALSE	Set automatic status if RMOT-input is in local
M14_Spare	BOOL	FALSE	Spare
M15_Test	BOOL	FALSE	Test bit (1) (not used)

b) Time values (to adjust in PCONF in DB83 for each object)

Timer	Format	Preset	Description
DlyTActFault	TIME	T#0MS	Set point Action Fault delay
DlyTOpening	TIME	T#0MS	Set point Waiting Delay Slow->Fast
DlyTClosing	TIME	T#0MS	Set point Waiting Delay Fast->Slow
DlyTFeedback	TIME	T#0MS	Set point feedback fault delay
DlyTReverse	TIME	T#0MS	Set point change direction
DlyTOvertravel	TIME	T#0MS	Set point auto alarm delay
DlyTExtFault	TIME	T#0MS	Set point external fault delay
DlyTRestart	TIME	T#0MS	Set point restart after blackout delay

### 10.3 Pumps / Motors

#### 10.3.1 Used Resources

Resource	Description
FB80	Standard object function block
DB80	Record data block (includes all records of type "Motor 1")
FB180	Distribution function block for objects of type "Motor 1"
DB280	Instance data block to FB180
DB380	I/O Data to OS
FC89	Delay timer (subroutine for FB83)
SFC20	Block move /subroutine for FB80), block implemented in firmware of PLC

Used memory space [in byte]:

Resource	Local data	MC7	Load memory	Work memory
FB80	50	2390	3102	2426
DB80 (50 objects)		5200	12074	5236
DB80 (one add. object)		+104	+240	+104
FB180 (50 objects)	8	8220	18404	8256
FB180 (one add. object)	+0	min. +44	min. +246	min. +44
DB280 (50 objects)		6800	16880	6836
DB280 (one add. object)		+136	+336	+136
DB380 (50 objects)		500	1082	536
DB380 (one add. object)		+10	+22	+10
FC89	12	218	334	254
SFC20	-	-	-	-

**Hint:** Timers, counters and global memory are not used by the process control software.

#### 10.10.2 Description I/O Area



Figure: Block view of motor1 object with I/O interface

Name	Type	Format	Initial value
<b>OBJ</b>	<b>IN</b>	<b>INT</b>	<b>0</b>
Description			
Serial number of motor object in record DB. Used to point to the beginning of object record data set. Object Number Range is '1...n' for 'n' objects.			



Name	Type	Format	Initial value
<b>RMOT</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Remote' feedback from local panel of motor (process signal). 0 -> LOCAL 1 -> REMOTE Motor will only be indicated in local mode (no feedback control, no control access from OS). In remote mode the operator has the possibility to start/stop the motor.			

Name	Type	Format	Initial value
<b>RUN</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Running' feedback from motor (process signal). 0 -> Motor stopped 1 -> Motor running			

Name	Type	Format	Initial value
<b>FAUL</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'External fault input' (process signal). Please refer to the mode switch "ExtFaultNC" for the function of this input:  <u>ExtFaultNC = false:</u> 0 -> no fault 1 -> fault  <u>ExtFaultNC = true:</u> 0 -> fault 1 -> no fault			

Name	Type	Format	Initial value
<b>BLKO</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Blackout' input (process signal). 0 -> No Blackout 1 -> Blackout			

Name	Type	Format	Initial value
<b>ACOM</b>	<b>IN</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Auto commando interface. This interface is only active in REMOTE and AUTO mode (this is an PLC internal interface)			
Bit	Value	Function	Description
0	1	BlackoutStart	Restart motor after blackout
1	2	Standbyactive	Standby active (if a standby unit is connected always true)
2	4		
3	8		
4	16		
5	32		
6	64		
7	128	FetchData	Read request data from standby unit
8	256	Start	Command Start
9	512	Stop	Command Stop
10	1024	Failure	Failure from Automatic
11	2048	Lock	Lock Start/Stop commands
12	4096	Emergency off	Emergency shut down
13	8192		
14	16384		
15	32768		

Name	Type	Format	Initial value
<b>PCONF</b>	<b>IN</b>	<b>STRUC</b>	<b>0</b>
Description			
Record pointer. Do not change this input without consulting Siemens!			

Name	Type	Format	Initial value
<b>MSTA</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
'Start' output to motor (process signal). 0 -> no command to motor 1 -> Start command to motor			

Name	Type	Format	Initial value
<b>MSTO</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
'Stop' output to motor (process signal). 0 -> no command to motor 1 -> Stop command to motor			

Name	Type	Format	Initial value
<b>ASTAT</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Auto status interface, also active in non automatic states (this is an PLC internal interface)			
Bit	Function	Description	
0			
1			
2			
3			
4			
5			
6			
7			
8	Running	Motor status: Running	
9	Local	Motor status: Local	
10	Manual	Motor status: Manual	
11	Auto	Motor status: Auto	
12	Failure	Motor status: Failure	
13	Blackout	Motor status: Blackout	
14	Starting	Motor status: Starting	
15	Restarting	Motor status: Restarting	

Name	Type	Format	Initial value
<b>OSSTAT</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Status word to OS icon (internal interface). Connect this output to the interface of your OS.			
Value	Status		
0	Undefined		
1	Running		
2	Stopped		
3	Starting		
4	Standby		
5	Blackout		
+256	LOCAL Mode (Offset Value)		
+512	MANUAL Mode (Offset Value)		
+1024	AUTO Mode (Offset Value)		

Name	Type	Format	Initial value
<b>OSALARM</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Alarm word to OS icon (internal interface). Connect this output to the interface of your OS.			
Value	Status		
0	Undefined		
1	Normal		
2	External Fault Alarm		
3	Auto failure alarm		
4	Action alarm		
5	Non -Feedback alarm		
6	Lock		
7	Emergency OFF		
8	Blackout		

Name	Type	Format	Initial value
<b>OSCOM</b>	<b>IN/OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Command word from OS icon (internal interface). Connect this output to the interface of your OS.			
Bit	Value	Function	Description
0	1	Sart	Start command to Motor
1	2	Stop	Stop command to Motor
2	4		
3	8		
4	16		
5	32		
6	64		
7	128	Reset	Reset command to Motor
8	256		
9	512	MANUAL-Mode	MANUAL-Mode Command to Motor
10	1024	AUTO-Mode	AUTO-Mode Command to Motor
11	2048		
12	4096		
13	8192		
14	16384		
15	32768		

### 10.3.3 Description Static Variables (options)

a) Mode switches (to adjust in PCONF in DB80 for each object)

Mode switch	Format	Preset	Description
sw01	BOOL	FALSE	
ExtFaultNC	BOOL	FALSE	Mode switch External fault NC
sw03	BOOL	FALSE	
sw04	BOOL	FALSE	
sw05	BOOL	FALSE	
sw06	BOOL	FALSE	
sw07	BOOL	FALSE	
ConstOutput	BOOL	FALSE	Mode switch Constant Output (FALSE=Double Acting, TRUE=Single Act.)
ExtraAutomode	BOOL	FALSE	Extra Auto mode available
NoActAFeedbackA	BOOL	FALSE	Mode switch No action and feedback alarm
StopIfA	BOOL	FALSE	Mode switch Stop if alarm
RestrtAftBlack	BOOL	FALSE	Mode switch Restart after Blackout
RstAStartStop	BOOL	FALSE	Mode switch Reset Alarm with Start /Stop
CmdTdly	BOOL	FALSE	Mode switch Command time delay
LockBlockCmd	BOOL	FALSE	Mode switch Lock blocks commands

b) Time values (to adjust in PCONF in DB80 for each object)

Timer	Format	Preset	Description
DlyTCmd	TIME	T#0MS	Set point command time delay
DlyTBlackout	TIME	T#0MS	Set point blackout time delay
DlyTExtFault	TIME	T#0MS	Set point external fault delay
DlyTFeedback	TIME	T#0MS	Set point feedback fault delay
DlyTStrtActFault	TIME	T#0MS	Set point start action delay
DlyTAutoAl	TIME	T#0MS	Set point auto alarm delay
DlyTRestart	TIME	T#0MS	Set point restart after blackout delay
DlyTStpActFault	TIME	T#0MS	Set point stop action fault

### 10.4 Pumps / Motors (2 Speeds OR 2 Directions)

#### 10.4.1 Used Resources

Resource	Description
FB81	Standard object function block
DB81	Record data block (includes all records of type "Motor 2")
FB181	Distribution function block for objects of type "Motor 2"
DB281	Instance data block to FB181
DB381	I/O Data to OS
FC89	Delay timer (subroutine for FB81)
SFC20	Block move /subroutine for FB81), block implemented in firmware of PLC

Used memory space [in byte]:

Resource	Local data	MC7	Load memory	Work memory
FB81	50	3168	3904	3204
DB81 (25 objects)		2700	6258	2736
DB81 (one add. object)		+108	+248	+108
FB181 (25 objects)	8	3504	8838	3540
FB181 (one add. object)	+0	min. +44	min. +254	min. +44
DB281 (25 objects)		3550	8830	3586
DB281 (one add. object)		+142	+350	+142
DB381 (25 objects)		250	582	286
DB381 (one add. object)		+10	+20	+10
FC89	12	218	334	254
SFC20	-	-	-	-

**Hint:** Timers, counters and global memory are not used by the process control software.

#### 10.4.2 Description I/O Area



Figure: Block view of motor2 object with I/O interface

Name	Type	Format	Initial value
<b>OBJ</b>	<b>IN</b>	<b>INT</b>	<b>0</b>
Description			
Serial number of motor object in record DB. Used to point to the beginning of object record data set. Object Number Range is '1...n' for 'n' objects.			



Name	Type	Format	Initial value
<b>RMOT</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Remote' feedback from local panel of motor (process signal). 0 -> LOCAL 1 -> REMOTE Motor will only be indicated in local mode (no feedback control, no control access from OS). In remote mode the operator has the possibility to start/stop the motor.			

Name	Type	Format	Initial value
<b>RUN1</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Running' feedback from motor (process signal). 0 -> Motor not running in low speed /forward direction 1 -> Motor is running in low speed /forward direction			

Name	Type	Format	Initial value
<b>RUN2</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Running' feedback from motor (process signal). 0 -> Motor not running in high speed /reverse direction 1 -> Motor is running in high speed /reverse direction			

Name	Type	Format	Initial value
<b>FAUL</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
External fault input (process value). 0 -> no fault 1 -> fault			

Name	Type	Format	Initial value
<b>ACOM</b>	<b>IN</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Auto commando interface. This interface is only active in REMOTE and AUTO mode (this is an PLC internal interface)			
Bit	Value	Function	Description
0	1	BlackoutStart	Restart motor after blackout
1	2		
2	4		
3	8		
4	16		
5	32		
6	64		
7	128	FetchData	Read request data from standby unit
8	256	Start 1	Command Start (low speed / forward)
9	512	Stop	Command Stop
10	1024	Failure	Failure from Automatic
11	2048	Lock	Lock Start/Stop commands
12	4096	Emergency off	Emergency shut down
13	8192	Start 2	Command Start (high speed / reverse)
14	16384		
15	32768		

Name	Type	Format	Initial value
<b>BLKO</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
Blackout indication (process value). 0 -> No Blackout 1 -> Blackout			

Name	Type	Format	Initial value
<b>PCONF</b>	<b>IN</b>	<b>STRUC</b>	<b>0</b>
Description			
Record pointer. Do not change this input without consultation Siemens!			

Name	Type	Format	Initial value
<b>MST1</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
'Start' output to motor (process signal). 0 -> no command to motor 1 -> Start low/forward command to motor			

Name	Type	Format	Initial value
<b>MST2</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
'Start' output to motor (process signal). 0 -> no command to motor 1 -> Start high/reverse command to motor			

Name	Type	Format	Initial value
<b>MSTO</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
'Stop' output to motor (process signal). 0 -> no command to motor 1 -> Stop command to motor			

Name	Type	Format	Initial value
<b>STAT</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Auto status interface, also active in non automatic states (this is an PLC internal interface)			
Bit	Function	Description	
0	Running2	Motor status: Running 'High/Reverse'	
1	Starting2	Motor status: Starting 'High/Reverse'	
2			
3			
4			
5			
6			
7			
8	Running1	Motor status: Running 'Low/Forward'	
9	Local	Motor status: Local	
10	Manual	Motor status: Manual	
11	Auto	Motor status: Auto	
12	Failure	Motor status: Failure	
13	Blackout	Motor status: Blackout	
14	Starting1	Motor status: Starting 'Low/Forward'	
15	Restarting	Motor status: Restarting	

Name	Type	Format	Initial value
<b>OSSTAT</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Status word to OS icon (internal interface). Connect this output to the interface of your OS.			
Value	Status		
0	Undefined		
1	Running1 'Low / Forward'		
2	Running2 'High / Reverse'		
3	Stopped		
4	Starting1 'Low / Forward'		
5	Starting2 'High / Reverse'		
6			
7	Blackout		
+256	LOCAL Mode (Offset Value)		
+512	MANUAL Mode (Offset Value)		
+1024	AUTO Mode (Offset Value)		

Name	Type	Format	Initial value
<b>OSALARM</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Alarm word to OS icon (internal interface). Connect this output to the interface of your OS.			
Value	Status		
0	Undefined		
1	Normal		
2	External Fault Alarm		
3	Auto failure Alarm		
4	Action Alarm		
5	Non -Feedback Alarm		
6	Lock		
7	Emergency OFF		
8	Blackout		

Name		Type	Format	Initial value
<b>OSCOM</b>		<b>IN/OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description				
Command word from OS icon (internal interface). Connect this output to the interface of your OS.				
Bit	Value	Function	Description	
0	1	Start1	Start 'Low / Forward' command to Motor	
1	2	Start2	Start 'High / Reverse' command to Motor	
2	4	Stop	Stop command to Motor	
3	8			
4	16			
5	32			
6	64			
7	128	Reset	Reset command to Motor	
8	256			
9	512	MANUAL-Mode	MANUAL-Mode Command to Motor	
10	1024	AUTO-Mode	AUTO-Mode Command to Motor	
11	2048			
12	4096			
13	8192			
14	16384			
15	32768			

### 10.4.3 Description Static Variables (options)

a) Mode switches (to adjust in PCONF in DB81)

Mode switch	Format	Preset	Description
M0_ConstantOut	BOOL	FALSE	Constant Output MST1 MST2 MSTO (1); Feedback depended Output Anst. (0)
M1_AutoManLoc	BOOL	FALSE	AUTO/ MANUAL/ LOCAL (1); MANUAL/ LOCAL (0)
M2_NoActFeedFault	BOOL	FALSE	No Action- and Non-Feedback fault(1),
M3_NoStopIfFault	BOOL	FALSE	No Stop-Signal if Failure occurs (1)
M4_RestrAftBlack	BOOL	FALSE	Restart after Blackout (1)
M5_ResetOSonly	BOOL	FALSE	Reset Alarm only with RESET from OS (1); RESET and START/STOP-Command (0)
M6_CmdTimeDly	BOOL	FALSE	Command Time Delay active (1)
M7_Chg12viaStop	BOOL	FALSE	Change from 1->2 via STOP(1)
M8_Chg21viaStop	BOOL	FALSE	Change from 2->1 via STOP(1)
M9_StopIfAutoFail	BOOL	FALSE	NO AVAIL Direct Stop if Auto Failure (1)
M10_LockBlockCmd	BOOL	FALSE	Lock blocks commands
sw11	BOOL	FALSE	
sw12	BOOL	FALSE	
sw13	BOOL	FALSE	
sw14	BOOL	FALSE	
sw15	BOOL	FALSE	

b) Time values (to adjust in PCONF in DB81)

Timer	Format	Preset	Description
DlyTCmd	TIME	T#0MS	Set point Command time delay
DlyTBlackout	TIME	T#0MS	Set point blackout time delay
DlyTExtFault	TIME	T#0MS	Set point external fault delay
DlyTFeedback	TIME	T#0MS	Set point feedback fault delay
DlyTActFault	TIME	T#0MS	Set point Action Fault delay
DlyTAutoAl	TIME	T#0MS	Set point auto alarm delay
DlyTRestart	TIME	T#0MS	No AVAIL Set point restart after blackout delay
DlyTChg12	TIME	T#0MS	Set point Waiting Delay 1->2
DlyTChg21	TIME	T#0MS	Set point Waiting Delay 2->1



### 10.5 Pumps / Motors (2 Speeds AND 2 Directions)

#### 10.5.1 Used Resources

Recourse	Description
FB82	Standard object function block
DB82	Record data block (includes all records of type "Motor 3")
FB182	Distribution function block for objects of type "Motor 3"
DB282	Instance data block to FB182
DB382	I/O Data to OS
FC89	Delay timer (subroutine for FB82)
SFC20	Block move /subroutine for FB82), block implemented in firmware of PLC

Used memory space [in byte]:

Recourse	Local data	MC7	Load memory	Work memory
FB82	74	4788	5542	4824
DB82 (25 objects)		2600	6108	2636
DB82 (one add. object)		+104	+242	+104
FB182 (25 objects)	8	3504	8988	3540
FB182 (one add. object)	+0	min. +44	min. +246	min. +44
DB282 (25 objects)		3500	8930	3536
DB282 (one add. object)		+140	+354	+140
DB382 (25 objects)		250	582	286
DB382 (one add. object)		+10	+20	+10
FC89	12	218	334	254
SFC20	-	-	-	-

**Hint:** Timers, counters and global memory are not used by the process control software.

#### 10.5.2 Description I/O Area

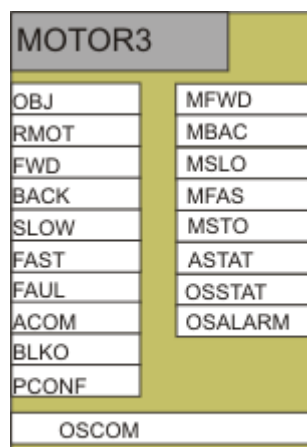


Figure: Block view of motor3 object with I/O interface

Name	Type	Format	Initial value
<b>OBJ</b>	<b>IN</b>	<b>INT</b>	<b>0</b>
<b>Description</b>			
Serial number of motor object in record DB. Used to point to the beginning of object record data set. Object Number Range is '1...n' for 'n' objects.			

Name	Type	Format	Initial value
<b>RMOT</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Remote' feedback from local panel of motor (process signal). 0 -> LOCAL 1 -> REMOTE Motor will only be indicated in local mode (no feedback control, no control access from OS). In remote mode the operator has the possibility to start/stop the motor.			

Name	Type	Format	Initial value
<b>FWD</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Running forward' feedback from motor (process signal). 0 -> Motor is not running in forward direction 1 -> Motor is running in forward direction			

Name	Type	Format	Initial value
<b>BACK</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Running reverse' feedback from motor (process signal). 0 -> Motor is not running in reverse direction 1 -> Motor is running in reverse direction			

Name	Type	Format	Initial value
<b>SLOW</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Running slow' feedback from motor (process signal). 0 -> Motor is not running in low speed 1 -> Motor is running in low speed			

Name	Type	Format	Initial value
<b>FAST</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Running fast' feedback from motor (process signal). 0 -> Motor is not running in high speed 1 -> Motor is running in high speed			

Name	Type	Format	Initial value
<b>FAUL</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
External fault input (process signal). Please refer to the mode switch "M10_ExtFaultNC" for the function of this input:			
<u>M10_ExtFaultNC = false:</u>			
0 -> no fault			
1 -> fault			
<u>M10_ExtFaultNC = true:</u>			
0 -> fault			
1 -> no fault			

Name	Type	Format	Initial value																																																																				
<b>ACOM</b>	<b>IN</b>	<b>WORD</b>	<b>W#16#0</b>																																																																				
<b>Description</b>																																																																							
Auto commando interface. This interface is only active in REMOTE and AUTO mode (this is an PLC internal interface)																																																																							
<table border="1"> <thead> <tr> <th>Bit</th><th>Value</th><th>Function</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>1</td><td>BlackoutStart</td><td>Restart motor after blackout</td></tr> <tr> <td>1</td><td>2</td><td></td><td></td></tr> <tr> <td>2</td><td>4</td><td></td><td></td></tr> <tr> <td>3</td><td>8</td><td></td><td></td></tr> <tr> <td>4</td><td>16</td><td></td><td></td></tr> <tr> <td>5</td><td>32</td><td></td><td></td></tr> <tr> <td>6</td><td>64</td><td></td><td></td></tr> <tr> <td>7</td><td>128</td><td>FetchData</td><td>Read request data from standby unit</td></tr> <tr> <td>8</td><td>256</td><td>Start forward slow</td><td>Command Start (forward and low speed)</td></tr> <tr> <td>9</td><td>512</td><td>Stop</td><td>Command Stop</td></tr> <tr> <td>10</td><td>1024</td><td>Failure</td><td>Failure from Automatic</td></tr> <tr> <td>11</td><td>2048</td><td>Lock</td><td>Lock Start/Stop commands</td></tr> <tr> <td>12</td><td>4096</td><td>Emergency off</td><td>Emergency shut down</td></tr> <tr> <td>13</td><td>8192</td><td>Start forward high</td><td>Command Start (forward and high speed)</td></tr> <tr> <td>14</td><td>16384</td><td>Start back slow</td><td>Command Start (reverse and low speed)</td></tr> <tr> <td>15</td><td>32768</td><td>Start back fast</td><td>Command Start (reverse and high speed)</td></tr> </tbody> </table>				Bit	Value	Function	Description	0	1	BlackoutStart	Restart motor after blackout	1	2			2	4			3	8			4	16			5	32			6	64			7	128	FetchData	Read request data from standby unit	8	256	Start forward slow	Command Start (forward and low speed)	9	512	Stop	Command Stop	10	1024	Failure	Failure from Automatic	11	2048	Lock	Lock Start/Stop commands	12	4096	Emergency off	Emergency shut down	13	8192	Start forward high	Command Start (forward and high speed)	14	16384	Start back slow	Command Start (reverse and low speed)	15	32768	Start back fast	Command Start (reverse and high speed)
Bit	Value	Function	Description																																																																				
0	1	BlackoutStart	Restart motor after blackout																																																																				
1	2																																																																						
2	4																																																																						
3	8																																																																						
4	16																																																																						
5	32																																																																						
6	64																																																																						
7	128	FetchData	Read request data from standby unit																																																																				
8	256	Start forward slow	Command Start (forward and low speed)																																																																				
9	512	Stop	Command Stop																																																																				
10	1024	Failure	Failure from Automatic																																																																				
11	2048	Lock	Lock Start/Stop commands																																																																				
12	4096	Emergency off	Emergency shut down																																																																				
13	8192	Start forward high	Command Start (forward and high speed)																																																																				
14	16384	Start back slow	Command Start (reverse and low speed)																																																																				
15	32768	Start back fast	Command Start (reverse and high speed)																																																																				

Name	Type	Format	Initial value
<b>BLKO</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
Blackout indication (process signal).			
0 -> No Blackout			
1 -> Blackout			

Name	Type	Format	Initial value
<b>PCONF</b>	<b>IN</b>	<b>STRUC</b>	<b>0</b>
<b>Description</b>			
Record pointer. Do not change this input without consulting Siemens!			

Name	Type	Format	Initial value
<b>MFWD</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
'Start Forward' output to motor (process value). 0 -> no command to motor 1 -> 'Start forward' command to motor			

Name	Type	Format	Initial value
<b>MBAC</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
'Start Reverse' output to motor (process value). 0 -> no command to motor 1 -> 'Start reverse' command to motor			

Name	Type	Format	Initial value
<b>MSLO</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
'Start Slow' output to motor (process value). 0 -> no command to motor 1 -> 'Start Low Speed' command to motor			

Name	Type	Format	Initial value
<b>MFAS</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
'Start Fast' output to motor (process value). 0 -> no command to motor 1 -> 'Start High Speed' command to motor			

Name	Type	Format	Initial value
<b>MSTO</b>	<b>OUT</b>	<b>BOOL</b>	<b>FALSE</b>
Description			
Stop output to motor (process value). 0 -> no command to motor 1 -> Stop command to motor			

Name	Type	Format	Initial value
<b>STAT</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Auto status interface, also active in non automatic states (this is an PLC internal interface)			
Bit	Function	Description	
0	Running Fast Forward	Motor status: Running 'Fast Forward'	
1	Running Slow Reverse	Motor status: Starting 'Slow Reverse'	
2	Running Fast Reverse	Motor status: Starting 'Fast Reverse'	
3			
4			
5			
6			
7			
8	Running Slow Forward	Motor status: Running 'Slow Forward'	
9	Local	Motor status: Local	
10	Manual	Motor status: Manual	
11	Auto	Motor status: Auto	
12	Failure	Motor status: Failure	
13	Blackout	Motor status: Blackout	
14	Starting 1	Motor status: Starting	
15	Restarting	Motor status: Restarting	

Name	Type	Format	Initial value
<b>OSSTAT</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Status word to OS icon (internal interface). Connect this output to the interface of your OS.			
Value	Status		
0	Undefined		
1	Running1 'Low / Forward'		
2	Running2 'High / Forward'		
3	Running3 'Low / Reverse'		
4	Running4 'High / Reverse'		
5			
6	Starting1 'Low / Forward'		
7	Starting2 'High / Forward'		
8	Starting3 'Low / Reverse'		
9	Starting4 'High / Reverse'		
10	Stopped		
11	Blackout		
+256	LOCAL Mode (Offset Value)		
+512	MANUAL Mode (Offset Value)		
+1024	AUTO Mode (Offset Value)		

Name	Type	Format	Initial value
OSALARM	OUT	WORD	W#16#0
Description			
Alarm word to OS icon (internal interface). Connect this output to the interface of your OS.			
Value	Status		
0	Undefined		
1	Normal		
2	External Fault Alarm		
3	Auto failure Alarm		
4	Action Alarm		
5	Non -Feedback Alarm		
6	Lock		
7	Emergency OFF		
8	Blackout		
Alarm word to OS icon (internal interface). Connect this output to the interface of your OS.			

Name	Type	Format	Initial value
<b>OSCOM</b>	<b>IN/OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Command word from OS icon (internal interface). Connect this output to the interface of your OS.			
Bit	Value	Function	Description
0	1	Start1	Start 'Low Forward' command to Motor
1	2	Start2	Start 'High Forward' command to Motor
2	4	Start3	Start 'Low Reverse' command to Motor
3	8	Start4	Start 'High Reverse' command to Motor
4	16	Stop	Stop command to Motor
5	32		
6	64		
7	128	Reset	Reset command to Motor
8	256		
9	512	MANUAL-Mode	MANUAL-Mode Command to Motor
10	1024	AUTO-Mode	AUTO-Mode Command to Motor
11	2048		
12	4096		
13	8192		
14	16384		
15	32768		



### 10.5.3 Description Static variables (options)

a) Mode switches (to adjust in PCONF in DB82 for each object)

Mode switch	Format	Preset	Comment
M0_ConstantOut	BOOL	FALSE	Constant Output MST1 MST2 MSTO (1); Feedback depended Output Anst. (0)
M1_AutoManLoc	BOOL	FALSE	AUTO/ MANUAL/ LOCAL (1); MANUAL/ LOCAL (0)
M2_NoActFeedFault	BOOL	FALSE	No Action- and Non-Feedback fault(1),
M3_NoStopIfFault	BOOL	FALSE	No Stop-Signal if Failure occurs (1)
M4_RestrAftBlack	BOOL	FALSE	Restart after Blackout (1)
M5_ResetOSonly	BOOL	FALSE	Reset Alarm only with RESET from OS (1); RESET and START/STOP-Command (0)
sw06	BOOL	FALSE	
M7_ChgSFviaStop	BOOL	FALSE	Change from Slow->Fast via STOP(1)
M8_ChgFSviaStop	BOOL	FALSE	Change from Fast->Slow via STOP(1)
M9_StopIfAutoFail	BOOL	FALSE	Direct Stop if Auto Failure (1)
M10_ExtFaultNC	BOOL	FALSE	External Fault is NC (1)
sw11	BOOL	FALSE	
sw12	BOOL	FALSE	
sw13	BOOL	FALSE	
sw14	BOOL	FALSE	
sw15	BOOL	FALSE	

b) Time values (to adjust in PCONF in DB82 for each object)

Timer	Format	Preset	Description
DlyTExtFault	TIME	T#0MS	Set point external fault delay
DlyTFeedback	TIME	T#0MS	Set point feedback fault delay
DlyTActFault	TIME	T#0MS	Set point Action Fault delay
DlyTAutoAI	TIME	T#0MS	Set point auto alarm delay
DlyTRestart	TIME	T#0MS	Set point restart after blackout delay
DlyTChgSF	TIME	T#0MS	Set point Waiting Delay Slow->Fast
DlyTChgFS	TIME	T#0MS	Set point Waiting Delay Fast->Slow
DlyTChgDirec	TIME	T#0MS	Set point change direction

### 10.6 Stand-By Pumps / Motors

#### 10.6.1 Used Resources

Resource	Description
FB84	Standard object function block
DB84	Record data block (includes all records of type "Standby")
FB184	Distribution function block for objects of type "Standby"
DB284	Instance data block to FB184
DB384	I/O Data to OS
FB90	Pulse timer (subroutine for FB84)
DB90	Instance data block to FB90
FC89	Delay timer (subroutine for FB84)
SFC20	Block move /subroutine for FB84), block implemented in firmware of PLC

Used memory space [in byte]:

Resource	Local data	MC7	Load memory	Work memory
FB84	74	2258	3036	2294
DB84 (10 objects)		740	1922	776
DB84 (one add. object)		+74	+184	+74
FB184 (10 objects)	8	1724	3600	1760
FB184 (one add. object)	+0	min. +44	min. +222	min. +44
DB284 (10 objects)		1100	2950	1136
DB284 (one add. object)		+110	+286	+110
DB384 (10 objects)		140	362	176
DB384 (one add. object)		+14	+24	+14
FB90	86	1456	1774	1492
DB90		68	242	104
FC89	12	218	334	254
SFC20	-	-	-	-

**Hint:** Timers, counters and global memory are not used by the process control software.

#### 10.6.2 Description I/O Area

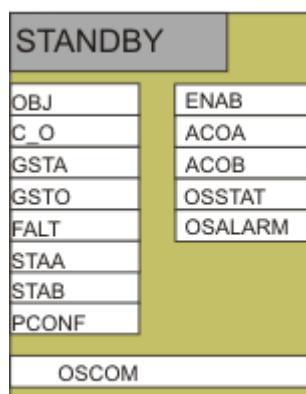


Figure: Block view of standby object with I/O interface

Name	Type	Format	Initial value
<b>OBJ</b>	<b>IN</b>	<b>INT</b>	<b>0</b>
<b>Description</b>			
Serial number of motor object in record DB. Used to point to the beginning of object record data set. Object Number Range is '1...n' for 'n' objects.			

Name	Type	Format	Initial value
<b>C_O</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Change Over' request from extern (hardware switch / program control) 0 -> Nothing to do 1 -> Change Over			

Name	Type	Format	Initial value
<b>GSTA</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Group start' command from extern (hardware switch / program control) 0 -> Nothing to do 1 -> Group start (start one motor, the other will be in standby)			

Name	Type	Format	Initial value
<b>GSTO</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
'Group stop' command from extern (hardware switch / program control) 0 -> Nothing to do 1 -> Group stop (stop both motors)			

Name	Type	Format	Initial value
<b>FALT</b>	<b>IN</b>	<b>BOOL</b>	<b>FALSE</b>
<b>Description</b>			
External fault input (process signal) 0 -> no fault 1 -> fault			

Name	Type	Format	Initial value
STAA / STAB	IN	WORD	W#16#0
Description			
Auto status words from both motors. STAA: Auto Status word from motor A. STAB: Auto Status word from motor B.			
Bit	Function	Description	
0			
1			
2			
3			
4			
5			
6			
7			
8	Running	Status Running	
9	Local	Status Local	
10	Manual	Status Manual	
11	Auto	Status Auto	
12	Failure	Status Failure	
13	Blackout	Status Blackout	
14	Starting	Status Starting	
15	Restarting	Status Restarting	

Please observe, that in a STANDBY configuration increased reliability is required. Therefore it is necessary to run the two instances of the MOTOR1 FB in two different PCUs. The STANDBY block can then be run in any of the two PCUs. The data from / to the 'remote' instance of MOTOR1 then have to be transmitted on a communication link (standard Simatic engineering, no code being generated by IMAC L for this link. This usually includes the STAx and ACOx word.

Name	Type	Format	Initial value
<b>PCONF</b>	<b>IN</b>	<b>STRUC</b>	<b>0</b>
Description			
Record pointer. Do not change this input without consultation Siemens!			

Name	Type	Format	Initial value
<b>ENAB</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Not used			

Name	Type	Format	Initial value
<b>ACOA / ACOB</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Auto Command words to both motors. ACOA: Auto Command word to motor A. ACOB: Auto Command word to motor B.			
Bit	Function	Description	
0	BlackoutStart	Blackout Start	
1	Standbyactive	Standby active (always = true)	
2			
3			
4			
5			
6			
7	FetchData	request data transfer from motor object	
8	Start	Start command to motor A/B	
9	Stop	Stop command to motor A/B	
10	Failure	External failure to motor A/B object	
11	Lock	Lock state to motor A/B	
12	Emergency off	Emergency off to motor A/B	
13			
14			
15			

Please observe, that in a STANDBY configuration increased reliability is required. Therefore it is necessary to run the two instances of the MOTOR1 FB in two different PCUs. The STANDBY block can then be run in any of the two PCUs. The data from / to the 'remote' instance of MOTOR1 then have to be transmitted on a communication link (standard Simatic engineering, no code being generated by IMAC L for this link. This usually includes the STAx and ACOx word.

Name	Type	Format	Initial value
<b>OSSTAT</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Not bits used			

Name	Type	Format	Initial value
<b>OSALARM</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Not bits used			

Name	Type	Format	Initial value
<b>OSCOM</b>	<b>IN/OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Command word from OS icon (internal interface). Connect this output to the interface of your OS.			
Bit	Value	Function	Description
0	1	Change Over	Change Over command to Motor pair
1	2	Stop	Stop command to Motor pair
2	4		
3	8		
4	16		
5	32		
6	64		
7	128		
8	256		
9	512		
10	1024		
11	2048		
12	4096		
13	8192		
14	16384		
15	32768		

### 10.6.3 Description Static variables (options)

a ) Mode switches (to adjust in DB84 for each object)

No Mode switches to adjust.

b) Time values (to adjust in PCONF in DB84 for each object)

Timer	Format	Preset	Description
DlyTCODelay	TIME	T#0MS	Set point Action Fault delay
DlyTStartingA	TIME	T#0MS	Set point Waiting Delay Slow->Fast
DlyTStartingB	TIME	T#0MS	Set point Waiting Delay Fast->Slow
DlyTFaultA	TIME	T#0MS	Set point feedback fault delay
DlyTFaultB	TIME	T#0MS	Set point change direction



### 10.7 Tanks

#### 10.7.1 Used Resources

Resource	Description
FB85	Standard object function block
DB85	Record data block (includes all records of type "Tank")
FB185	Distribution function block for objects of type "Tank"
DB285	Instance data block to FB185
DB385	I/O Data to OS
FC114	Norm value via curve (IMAC L AbvSpu)
FC166	Get status from external object (IMAC L AbvSpu)
FC167	Get value from external object (IMAC L AbvSpu)
SFC20	Block move /subroutine for FB85), block implemented in firmware of PLC

Used memory space [in byte]:

Resource	Local data	MC7	Load memory	Work memory
FB85	92	998	1348	1034
DB85 (10 objects)		440	1338	476
DB85 (one add. object)		+44	+132	+44
FB185 (10 objects)	8	1764	3490	1800
FB185 (one add. object)	+0	min. +44	min. +208	min. +44
DB285 (10 objects)		960	2660	996
DB285 (one add. object)		+96	+258	+96
DB385 (10 objects)		320	582	356
DB385 (one add. object)		+32	+50	+32
SFC20	-	-	-	-

#### 10.7.2 Description I/O Area

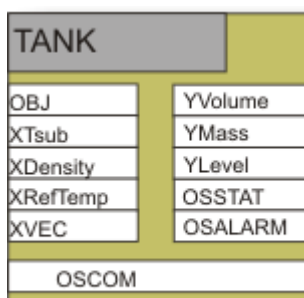


Figure: Block view of tank object with I/O interface

Name	Type	Format	Initial value
OBJ	IN	INT	0
Description			
Serial number of motor object in record DB. Used to point to the beginning of object record data set. Object Number Range is '1...n' for 'n' objects.			

Name	Type	Format	Initial value
<b>XTsub</b>	<b>IN</b>	<b>REAL</b>	<b>0.0</b>
Description			
Temperature substitution value. Only necessary for tanks with density compensation (please refer to mode switch "M1_TemperatureComp"). Value from OS or program. The Substitution value will be used, if the measured value is not available (i.e. Out of Range, Undefined, Faded Out or invalid Object Number specified for temperature Measurement -> parameter 'PCONF.ObjTemp'). The unit of this temperature is [°C].			

Name	Type	Format	Initial value
<b>XDensity</b>	<b>IN</b>	<b>REAL</b>	<b>0.0</b>
Description			
Density of tank medium. OS value (set point) or program control. The unit of this density is [kg/m3] or [tons/m3] (which is both the same value)..			

Name	Type	Format	Initial value
<b>XRefTemp</b>	<b>IN</b>	<b>REAL</b>	<b>0.0</b>
Description			
Reference temperature for given density 'XDensity'. OS value (set point) or program control. Only necessary for tanks with density compensation (please refer to mode switch "M1_TemperatureComp"). Value from OS or program. The unit of this temperature is [°C].			

Name	Type	Format	Initial value
<b>XVEC</b>	<b>IN</b>	<b>REAL</b>	<b>0.0</b>
Description			
Volume expansion coefficient. OS value (set point) or program control. Specifies the change of density with each Kelvin of temperature change, starting from density value XDensity at temperature XRefTemp. Only necessary for tanks with density compensation (please refer to mode switch "M1_TemperatureComp"). Value from OS or program. The unit of the Volume Expansion Coefficient is [1/K].			

Name	Type	Format	Initial value
<b>YVolume</b>	<b>OUT</b>	<b>REAL</b>	<b>0.0</b>
Description			
Computed Volume of the medium in the tank. The unit of the result is [m³].			

Name	Type	Format	Initial value
<b>YMass</b>	<b>OUT</b>	<b>REAL</b>	<b>0.0</b>
Description			
Computed Mass of the medium in the tank. The unit of the result is [Tons].			

Name	Type	Format	Initial value
<b>YLevel</b>	<b>OUT</b>	<b>REAL</b>	<b>0.0</b>
Description			
Computed Level of the medium in the Tank. The unit of the result is [m].			

Name	Type	Format	Initial value
<b>OSSTAT</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Not used.			

Name	Type	Format	Initial value
<b>OSALARM</b>	<b>OUT</b>	<b>WORD</b>	<b>W#16#0</b>
Description			
Not used.			

### 10.7.3 Description Static Variables (options)

a) Mode switches (to adjust in PCONF in DB85 for each object)

M0_SensorConfiguration	BOOL	FALSE	0: Pressure Sensor; 1: Level Sensor
M1_TemperatureComp	BOOL	FALSE	0: without Compensation; 1: with compensation
M2	BOOL	FALSE	
M3	BOOL	FALSE	
M4	BOOL	FALSE	
M5	BOOL	FALSE	
M6	BOOL	FALSE	
M7	BOOL	FALSE	
M8	BOOL	FALSE	
M9	BOOL	FALSE	
M10	BOOL	FALSE	
M11	BOOL	FALSE	
M12	BOOL	FALSE	
M13	BOOL	FALSE	
M14	BOOL	FALSE	
M15	BOOL	FALSE	

b) Values (to adjust in PCONF in DB85)

Conf. Value	Format	Preset value	Description
ObjInput	INT	0	AvEdit (IMAC L) object number of pressure/level transmitter A pressure transmitter have to deliver a result in the unit [bar]. A level transmitter has to deliver a result in the unit [m].
ObjTemp	INT	0	AvEdit (IMAC L) object number of temperature transmitter. This temperature transmitter has to deliver a result in the unit [°C].
SkIRefNo	BYTE	B#16#0	AvEdit (IMAC L) Scaling Group number, that contains the tank table X = Tank level in unit [m]. Y = Tank Volume in unit [m3]
Offset	REAL	0.000000e+000	Offset of pressure transmitter from bottom of tank in unit [m]
LL_XTsub	REAL	0.000000e+000	Low limit operator input tank temperature in unit [°C]
HL_XTsub	REAL	0.000000e+000	High limit operator input tank temperature in unit [°C]
LL_XDensity	REAL	0.000000e+000	Low limit operator input density (t ref) in unit [kg/dm <sup>3</sup> ]
HL_XDensity	REAL	0.000000e+000	High limit operator input density (t ref) in unit [kg/dm <sup>3</sup> ]
LL_XRefTemp	REAL	0.000000e+000	Low limit operator input reference temperature in unit [°C]
HL_XRefTemp	REAL	0.000000e+000	High limit operator input reference temperature in unit [°C]
LL_XVEC	REAL	0.000000e+000	Low limit operator input volume expansion coefficient [1/K]
HL_XVEC	REAL	0.000000e+000	High limit operator input volume expansion coefficient [1/K]

### 10.8 Converting SoftPLC7 Programs

SoftPLC7 programs in IMAC L are used e.g. for controlling the Extended Alarm System (EAS). SoftPLC7 is an application, that is capable to run a complete Simatic S7 program as a software task under windows.

The IMAC L Template project contains two SoftPLC7 programs, which are required to run IMAC L optional functions:

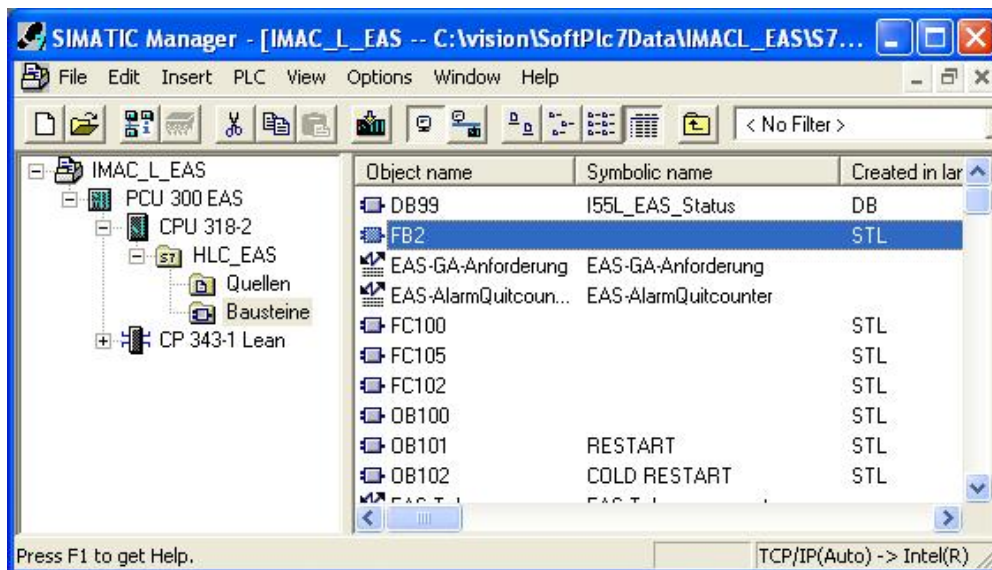
- EAS (Extended Alarm System) (path 'C:\vision\SoftPLC7Data\IMACL\_EAS')
- OS99 (for SIC functions) (path 'C:\vision\SoftPLC7Data\IMACL\_OS99')

The following section shows, how an existing Simatic Manager S7 project is converted for use with SoftPLC7. The Names, screenshots and directories used in this example are those of the EAS driver program of IMAC L.

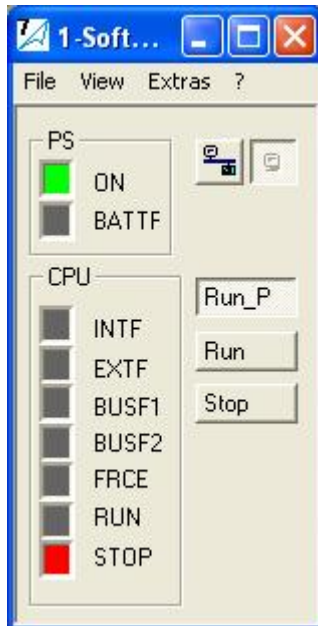
This section will not show, how to create / modify the existing S7 project - this is standard Simatic engineering.

To convert an existing Simatic S7 project for use with SoftPLC7:

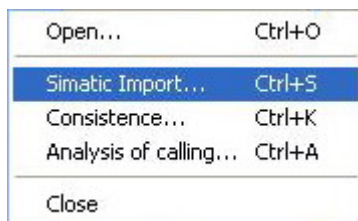
- Create or modify the Simatic S7 project as desired using the Simatic Manager application :



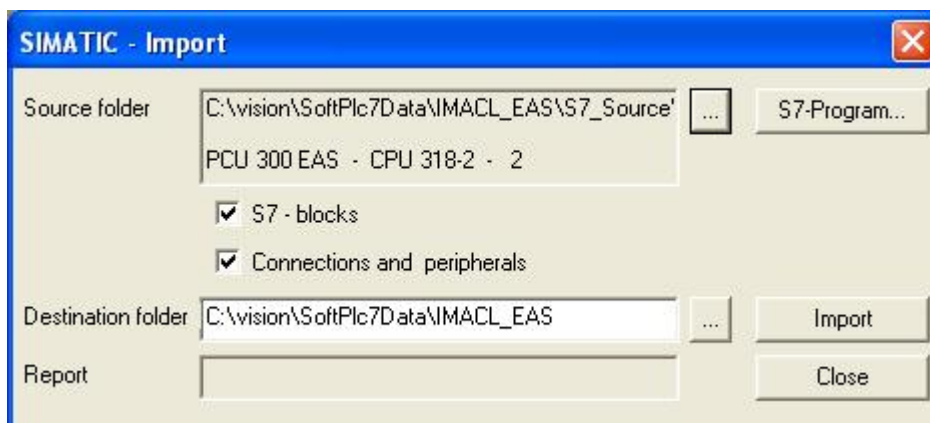
- Close the Simatic Manager Application
- Click on 'Start → Run'
- Enter 'C:\vision\bin\SoftPlc7.exe'
- Click on 'OK'
- SoftPLC7 will open with no program loaded:



- Click on 'File → Simatic Import'

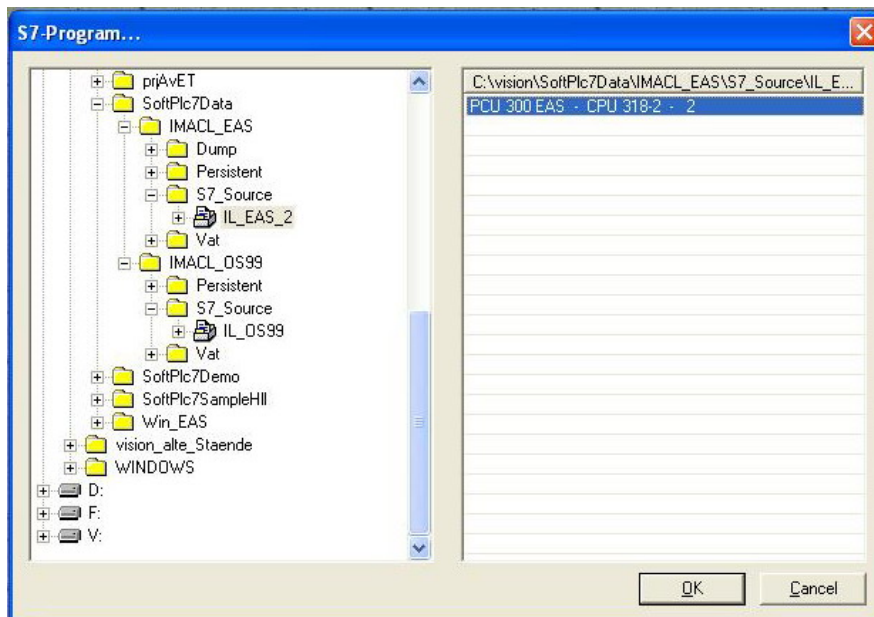


- An input window will prompt for the required paths for Importing :



- The 'Source folder' has to point to a subdirectory of the Simatic Manager directory where the S7 project that is to be converted is stored. Use the 'browse button' on the right side of the input field to locate your project. If you open the IMAC L EAS program from the path as specified at the start of this section, this should look like this :



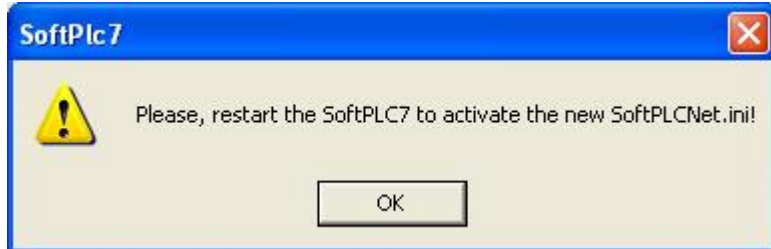


- Click on the Simatic Manager Project folder in the left window
- Click on the CPU name in the right window
- Click on 'OK' (→ back to 'Simatic Import' dialogue)
- Enter the 'Destination folder'. This can be any directory. For IMAC L all SoftPLC7 programs of a project are collected under the directory 'c:\vision\SoftPlc7Data'. You should have one subdirectory for each SoftPlc7 program. For EAS this directory is named IMACL\_EAS. If you use the 'browse button' on the right side of the input field, this should look like this :



- Click on the desired 'Destination' folder
- Click on 'OK'

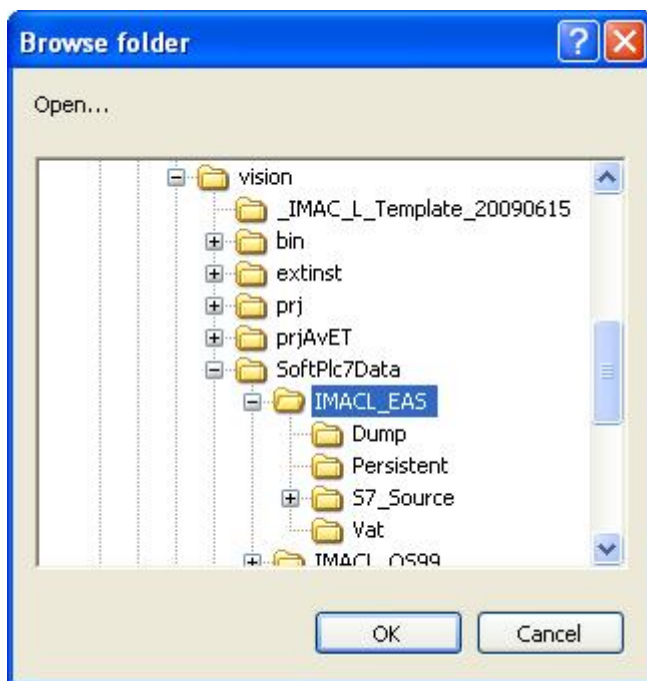
- Click on the 'Convert' button in the 'Convert...' window.
- Ignore the following PopUp-Window by clicking 'OK':



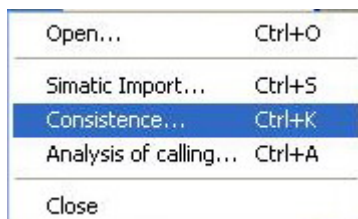
- Successful conversion will be reported as 'Ready' in the 'Report' field :



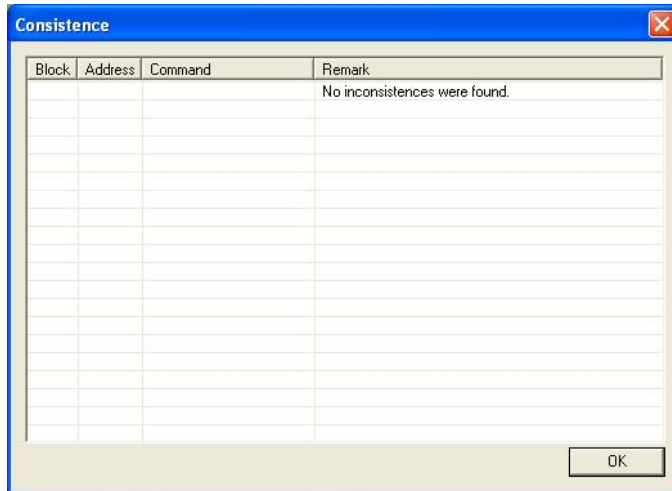
- Click on 'Close' in the 'Convert...' window.
- In the SoftPLC7 window click on 'File → Open'
- In the 'Browse Folder' window point to the target directory, where the SoftPLC7 program was created. For EAS this should be 'C:\vision\SoftPlc7Data\IMACL\_EAS' :



- Click on 'OK'
- In the SoftPLC7 window click on 'File → Consistence'



- A summary of inconsistencies will be displayed. If no inconsistencies were found, the 'Consistence' window will indicate : 'No inconsistencies were found.' :



- Click on 'OK'
- Close the SoftPlc7 window.

The SoftPlc7 program you have created is ready for use now. For creating a new SoftPlc7 program, a SoftPlc7.ini file has to be created and placed in the target directory. Refer to the AbHelp application for more information on how to define this file.

For IMAC L 'EAS' the corresponding SoftPlc7.ini file is part of the IMAC L template files. To run the EAS application (on one OS only!), the start batch file for IMAC L has to be modified to contain the EAS driver and SoftPlc7 program as well. Typically this will look like this :

```
rem --- UDP-Treiber und VISU ---
cd c:\vision\bin
start /min udp2av.exe c:\vision\bin\IMAC_L.do
start prcs IMAC_L /nosplash

rem --- SoftPlc7 (IMAC L - OS 99) ---
rem cd c:\vision\SoftPlc7Data\IMACL_OS99
rem start/min c:\vision\bin\SoftPlc7.exe
/inic=c:\vision\SoftPlc7Data\IMACL_OS99\SoftPlc7.ini /rw /nosplash

rem --- wait before starting EAS ---
c:
cd c:\vision\prj\avet
sleep 20

rem --- SoftPlc7 (IMAC L - EAS) ---
cd c:\vision\SoftPlc7Data\IMACL_EAS
start/min c:\vision\bin\SoftPlc7.exe /inic=c:\vision\SoftPlc7Data\IMACL_EAS\SoftPlc7.ini
/rw /nosplash

rem --- wait before starting SoftPLC EAS ---
c:
cd c:\vision\prj\avet
sleep 10

rem --- Driver for EAS System Communication ---
c:
cd c:\vision\bin
start/min c:\vision\bin\av2softeas.exe IMAC_L
```

**Hint:** It is recommended to store an archive version (zip file) of the Simatic Manager project that has been converted as a backup. Use the 'C:\vision\SoftPlc7Data' directory as a default. Modifications to the project have to be done via the Simatic Manager again. The conversion procedure has to be repeated completely afterwards. If you do not have the Simatic Manager project at hand, from which the SoftPlc7 program was created, you will not be able to apply changes to the SoftPlc7 program later on.

## 11. IMAC L Human Machine Interface (HMI)

### 11.1. Basic Rules for the IMAC L HMI System

There is a basic set of rules regarding the IMAC L HMI system :

- The Screen resolution of IMAC L Operator Stations is fixed at 1280 x 1024 pixels. Other resolutions are possible, but due to a different page ratio the display will be distorted (a circle will not be displayed as a circle any more).
- All colours and colour palettes in IMAC L are fixed ! Do not change the colours chosen for any process objects (figures, piping). The IMAC L colour palettes are designed to work with day/night screen design switching by swapping colour palettes. Changes to the predefined colours or colour palettes may yield unexpected results for the night design.
- Do not change the background colours! Background colours are part of the figure design in IMAC L. Changing the background colours will require to modify all existing process figures to be used (this is seriously **\*\*big\*\*** work) !

### 11.2. Tips on IMAC L Screen Size / Resolution issues

Basically IMAC L and its graphics tool Alpha-Vision uses vector graphics to display its graphics pictures. As a default any graphics picture drawn in any whatsoever resolution will be displayed in two possible ways: Either scaling the picture to the available window size (fit to screen) or displaying it in the 'original size' that was used to create the picture (no scaling). If the picture size used to create the picture precisely meets the available physical screen size, the result of both drawing methods for a full screen GD is identical.

The resolution of the physical screen for IMAC L is 1280 x 1024 pixel (physical PC screen)

The Application Area Setting (Layout Editor) is 1280 x 818 pixel (physical PC screen reduced by Header Area, Menu Area and Alarm Area)

The default Picture Size IMAC L uses is 1256 x 756 pixels (Application Area size reduced by the size of the two slider bars). This picture size will precisely meet the available physical screen area size, if the 'Application Area Caption Bar' is switched off.

#### 11.2.1. Caption Bar of the Application Area

The caption bar of the IMAC L application area can be switched ON or OFF. There are advantages and disadvantages related to the window caption bar being ON or OFF:

With the caption bar ON the caption bar will always indicate the name of the GD currently being displayed. In 'Split Screen' mode it will be easily visible, which window is receiving the user input (caption color). On the other hand the physical size of the screen is slightly smaller than the original picture size (as used for drawing it). Thus the GD will have to be drawn either in 'fit to screen' size (key CTRL-U) or in 'original size' (CTRL-O). For unzooming, it has to be distinguished, if in 'Split Screen' mode or not. In 'Split Screen' mode 'CTRL-U' will unzoom by fitting the GD into the window using the scaling required to display the whole GD into the small 'split' window. CTRL-U is using Alpha-Vision function 'Wnd\_UnZoom()'. In non-split mode 'CTRL-O' will unzoom by displaying the GD in its original size without any scaling by using the Alpha-Vision function 'Wnd\_OriginalSize()'. In this view the slider bars may be active, if the original picture size to be displayed is larger than the available physical screen area. If using 'CTRL-U' in non-split mode, the display may be slightly different from the default view from calling a picture from a button action.

With the caption bar OFF the available application area is not reduced by the caption bar size. The full size of 1256x756 pixels is available for the application. The physical size of the screen available for the application area meets the picture size used to create the pictures. Because of this, unzooming with 'CTRL-U' will always display a GD in the required way (original size, if display is full-screen or scaled down to the windows size, if in 'Split Screen' mode). As a drawback there will be no default indication showing, which GD is currently being displayed in the application area. If 'Split Screen' is used, a caption bar being switched OFF will make it more difficult to determine, which window is currently receiving the user input (i.e. has the windows focus). The slider bars will be active in 'zoom' mode only.

It is recommended in IMAC L to have the caption bar switched 'ON'. This setting is being used as a default setting for the IMAC L template files. Nevertheless it is up to each project to give it a try without the caption bar. The caption bar is being switched ON / OFF with the 'cfg.exe' application:

- Click on 'Start → Run'
- Enter 'C:\vision\bin\cfg.exe'
- Click on 'OK'
- In the 'Cfg' editor click on File → Open
- Point to the configuration file C:\vision\bin\IMAC\_L.cfg
- Click on 'OK'
- Click on the tab 'Display'
- Check / Uncheck field 'Displays Movable'. If this checkmark is set, the caption bar will be on. If it is unchecked, the caption bar is off.
- In the 'Cfg' editor click on File → Save
- Close the 'Cfg' editor
- Restart the OS

#### 11.2.1. Slider Bars of the Application Area

There is no setting to make the 'slider bars' in IMAC L disappear. The slider bars will always be visible. They are grayed (disabled), if no zooming is active and if no panning is required. The basic reason is the 'zoom screen' function and the 'original size' view in IMAC L. As soon as zooming is active, the slider bars will be active (not grayed) and will give the possibility to 'pan' around.

### 11.3. Tips for Printing with IMAC L

#### 11.3.1. Eventlog Printing with IMAC L

The windows name of the printer used for EventLog printing has to match the printer queue name referenced in GD 'X2'. The default name used in IMAC L is 'PRINTER'. The easiest way to connect an EVENTLOG printer is to simply rename the Windows printer used for EVENTLOG printing to the name 'PRINTER':

- Click on 'Start -> Settings -> Printers'.
- Click on the desired printer.
- Right-click -> Rename -> enter the new name 'PRINTER'.
- Press 'ENTER'.

The EventLog printing in the IMAC L template files is 'OFF' by default. It can be switched to 'ON' in picture X2 by pressing the button 'Event Print' on the lower left side of X2. A login with appropriate user rights is required for this, e.g. as the 'Super User' named 'SU'. Switching off again is by pressing 'EvPr Abort'.

Switching event printing to 'ON' by default is done via the Cfg.exe application:

- Click on 'Start → Run'
- Enter 'C:\vision\bin\cfg.exe'
- Click on 'OK'
- In the 'Cfg' editor click on File → Open
- Point to the configuration file C:\vision\bin\IMAC\_L.cfg
- Click on 'OK'
- Click on the tab 'Default'
- Set the checkmark 'Initial Event Printout'
- In the 'Cfg' editor click on File → Save
- Close the 'Cfg' editor
- Restart the OS

Event printing will collect new events until a whole page of events can be printed (standard windows page printer, line printing is not supported). A printout of a partially filled EventPrint page can be forced in GD 'X2' by pressing the 'Force Print' button.

The events currently pending for printing and those recently having been printed can be watched in a print layout style in the standard GD 'XELP'.

#### 11.3.2. Report Printing with IMAC L

If 'print' buttons are used e.g. in header menus to print a section data list, the following information applies:

- The default name for these 'Report' print functions in the IMAC L templates is 'PRINTER2'.
- It is recommended to rename the respective Windows printer to 'PRINTER2' to easily achieve this assignment.
- The name of the printer to be used for this type of printing is part of the 'Attributes Sequential' properties of each of the print buttons. The properties of these buttons have to be modified, if the printer name is not 'PRINTER2'.
- If, as a reaction to a button being pressed, a printout is meant to go to the same printer as used for the EventLog print, this printing action has to refer to a global system variable named '\_Queue'. This is basically the input variable that is behind the printer queue input field in GD X2. Thus '\_Queue' finally refers to the EventLog printer as specified in X2. An example on how to set this type of button up may be found in the menu area of GD 'XEL'.



The 'Attributes Sequential' of a button to print on the 'Report' printer named 'PRINTER2' looks like this (e.g. in header of the GD 'Section Data'):

The dialog box 'Edit attributes sequential [Button]' has a 'Category' list on the left with 'Action' selected. On the right are buttons: '<< Ok previous', 'Close', 'Ok, next >>', 'Cancel', and 'Undo'. The 'Action function' field contains 'Usr\_PrintPage(XSDP1,PRINTER2,0);Obj\_IncrData[\_XSDP,Val]'. The 'Description' field contains 'Start Print Section Data'. The 'Right mouse button action' field is empty. The 'Apply group' field is empty. The 'Shortcut' section has 'S' in the dropdown, and checkboxes for Ctrl (unchecked), Alt (checked), and Shift (unchecked). The 'Posttrigger function' field is empty.

The 'Attributes Sequential' of a button to print on the 'EVENTLOG' printer (printer naming from GD 'X2', standard = 'PRINTER') looks like this :

The dialog box 'Edit attributes sequential [Button]' has the same layout as the first one. The 'Action function' field contains 'SgPo\_Txt(\_QUEUE);Usr\_PushArgs();Usr\_PrintPage(XELP,0);'. The 'Description' field contains 'Start Print Event Log'. The 'Right mouse button action' field is empty. The 'Apply group' field is empty. The 'Shortcut' section has 'S' in the dropdown, and checkboxes for Ctrl (unchecked), Alt (checked), and Shift (unchecked). The 'Posttrigger function' field is empty.

## 11.4 IMAC L 'Hints & Tips' (HiTi)

Setting up Hints & Tips for MePos is achieved as follows:

### 1. Enter a HiTi-ID (6 characters) in the field 'HiTi' of selected or all MePos :

- See [→ 5.3.11 Field : 'HiTi' on page 40] for details on the HiTi-ID.
- The HiTi-ID is a freely selectable identifier for each HTML HiTi file (not the file name).
- No special Characters are allowed. Just 6 characters or digits.
- The 'HiTiID' refers to a HTML file that contains the desired help.
- Many MePos may refer to the same HiTi-ID.

### 2. Create all the HTML HiTi files needed (templates can be found on the IMAC L CD):

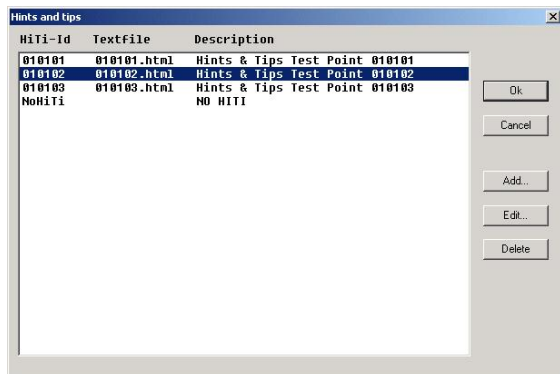
- For convenience use a HTML Editor of your choice to create / edit these files.
- Add the two 'AbHelp' specific comment tags to the header of each HiTi file. These special tags have to be placed immediately behind the initial HTML statement and will control the representation and display of the HiTi :

```
<HTML>
<!--ABHELP-KEYWORDS: 010102-->
<!--ABHELP-CONTENTENTRY:Folder 1\Folder 2\IMAC L Help on MePo 010102-->
<title>IMAC L MePo '010102' HiTi Test</title>
<BODY>
Testing IMAC L ! Point 010102 !
</BODY>
```

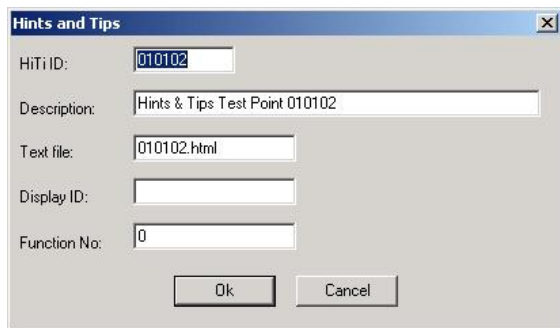
- The comment **<!--ABHELP-KEYWORDS: 010102-->** makes the direct link between the HiTi-ID '010102' (from MePo or GD) and the search results found in AbHelp.
- Multiple Keywords may be used, separated by a semicolon (";").
- The comment **<!--ABHELP-CONTENTENTRY:Folder 1\Folder2\IMAC L Help on ...** creates the entry / entries in the directory tree subdirectories of AbHelp.
- Tree sub-directories (e.g. 'Folder 1') may be created in multiple levels as required by just using them.
- The assortment in the resulting directories is alphabetical, but with 'folders first', if there are any sub-folders.
- If multiple HTML files will refer to the same 'CONTENTENTRY' folder, this folder will be created just once in the AbHelp tree and all objects referencing to this folder will be placed together in this folder. This means that the total tree structure in AbHelp is simply made up from all 'CONTENTENTRY' comments of all HTML files which AbHelp will find in the directory 'C:\vision\bin' during start of AbHelp.
- The HTML <title> tag is required to show the links in the 'Select topics' list (see below).
- Place all required HiTi HTML files in the directory 'C:\vision\bin\help'.
- Remove all pre-installed files in this directory to avoid, that the Alpha-Bit help files will be visible to the operator. You may either delete or better move them to a different directory of your choice.

### 3. Enter the references between HiTi-Ids and HTML file names in 'AbvEditor':

- Open the list 'PMU' → 'AbvBrowser' → 'File' → 'Hints and Tips'



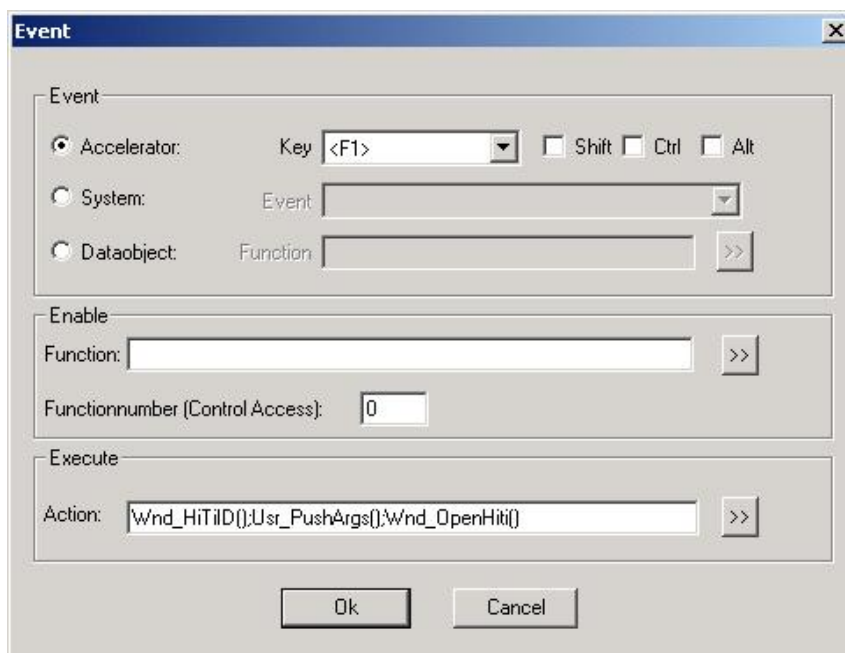
- Add one entry for each HTML HiTi file required



- These entries will provide the reference between HiTi-ID and HTML file name.

### 4. Activate 'Hiti' on pressing a global function key ('F1') (optional):

- Open the 'PMU' → 'AbvBrowser'
- Open 'File' → 'Global Events'
- Add or (if it exists edit) the Event for 'F1' to precisely the following settings:

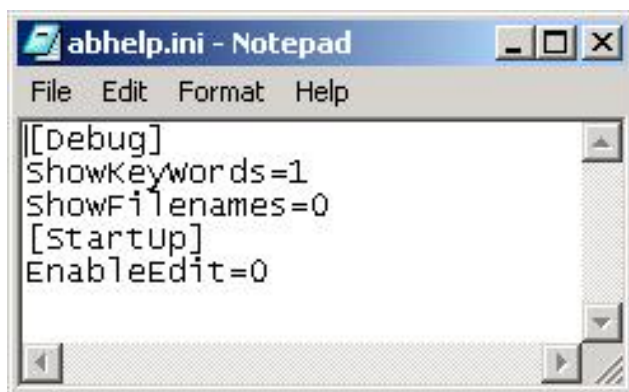


**5. Assign a HiTi-ID to some or all Graphics Pictures (optional):**

- Open the 'PMU' → 'AbvEditor'
- Open the Application Area of the GD with 'File' → 'Open'
- Click on 'Edit' → 'Information'
- In the 'dropdown box 'Hints & Tips' select the HiTi-ID to be assigned to this GD. Only those HiTi-Ids already entered in the AbvBrowser HiTi list are available here.
- Save your changes
- Pressing 'F1' in this GD will now directly link to the HTML file linked with this HiTi-ID.
- If 'F1' in a GD with no HiTi-ID assigned to it is pressed, IMAC L will prompt the operator to input the HiTi-ID manually. If he does not input anything, but just presses ENTER on an invalid or empty command prompt, AbHelp will open, but will just show its menu tree (no link to a specific HTML file). Manual navigation in AbHelp is always possible.

**6. Create a configuration file for AbHelp (optional)**

- The AbHelp application can optionally have its own configuration ('ini') file in the directory 'C:\vision\bin'.
- This file can be used to configure the indications and properties of AbHelp. A sample file of abhelp.ini can be found on the IMAC L CD :

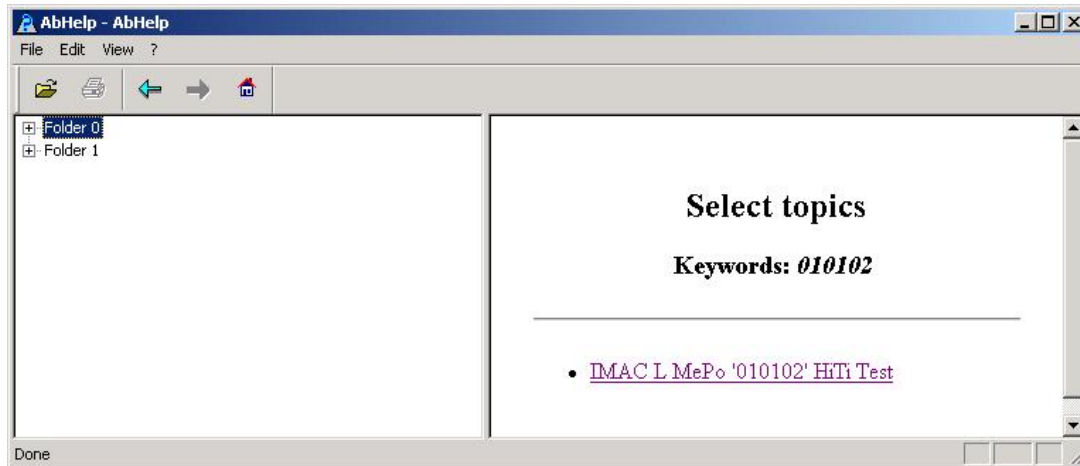


- The key '**ShowKeywords**' makes AbHelp show ('1') or hide ('0') the search keys used.
- The key '**ShowFileNames**' makes AbHelp show ('1') or hide ('0') the HTML file names.
- The key '**EnableEdit**' allows ('1') or disallows ('0') the user to modify the HTML files directly by using AbHelp controls.

**7. Recompile the System (at least from PMU on)**

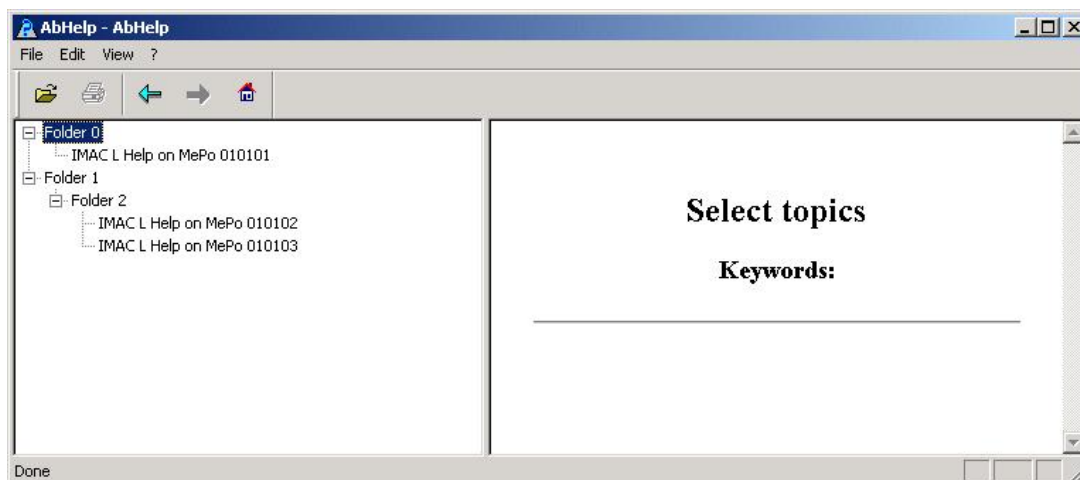
- After doing this, the 'HiTi' button in the Point Report GD ('XPR1') will directly link to the HTML files that have been created.
- HiTi is only available for the MePo type 'MP'. MePos of type 'PV' do not have HiTi.

The resulting display of AbHelp (from the Template files) will look like this (in this example called for the HiTi-ID '010102'):



The 'Select topics' list will show a list of all references in all AbHelp HTML files, which have the requested keyword in their 'KEYWORDS' comment tag. The text shown for the choices in this title list is taken from the HTML <title> tag. If this tag is missing, the selected topic will show up as a bullet point without a text, which will not be selectable. Therefore the <title> tag is always required.

For manual navigation in AbHelp, the resulting 'Tree style' display of AbHelp (from the Template files) will look like this:



The appearance / structure / naming of the tree directories can be modified by setting the 'special comment tag' of type 'CONTENTENTRY' in the HTML files (see above) appropriately.

If HiTi is called from a MePo (button 'HiTi' in POINT REPORT) or by pressing F1 in a GD, while there is no HiTi assigned to this specific object, the AbHelp will open with an empty 'Select Topics' list. The user will (as always) be allowed to manually navigate in the help tree.

## 11.5. User Rights and 'Station In Control' Concept

IMAC L offers a sophisticated system to assign and control user access to IMAC L. The basic concept for this is called STATION IN CONTROL. STATION IN CONTROL is a concept, which prevents, that a specific control function can be effected from more than one OS at a time. Additionally users will have to log in to the system. With this login, user rights are assigned to a user, which can vary from 'just acknowledge alarms' (watch keeper) to 'everything allowed' (super user). For more information on the 'user rights' and STATION IN CONTROL concept please refer to the IMAC L HMI description. The IMAC L acronym for STATION IN CONTROL is 'SIC'.

This section just describes the technical preconditions for SIC that have to be implemented in a system, if user rights / STATION IN CONTROL are required. Some examples show, how to assign user rights to SIC areas.

### 11.5.1. Start Batch to run SoftPLC Program for 'OS 99'

IMAC L requires a single source of information for SIC that is accessible to all IMAC L OSs. This source is required to determine the current 'STATION IN CONTROL' status. For this purpose, one instance of the SoftPLC7 simulation program is being loaded with a predefined configuration holding a set of process variables (PVs) for SIC. This predefined program usually does not require modification – except for AbvSPU updates (see below).

This 'OS 99' program must run on OS1 and may not run on any other OS. This program attaches to the Ethernet communication bus and can communicate with all OSs as if it were a real IMAC L PLC (station number 15). To start this program correctly, you will have to edit the start batch files being used to run IMAC L on the OSs in a way that they are different between the OS1 and the rest of the OSs:

The command to start the SoftPLC7 instance correctly from a batch file is as follows:

```
rem --- SoftPlc7 (IMAC L - OS 99) ---  
cd c:\vision\SoftPlc7Data\IMACL_OS99  
start/min C:\vision\bin\SoftPlc7.exe /ini=C:\Vision\SoftPlc7Data\IMACL_OS99\softPlc7.ini /rw
```

There is a version of the Start batch files (\*.cmd) in directory 'C:\vision\PrjAvEt\') which will automatically start the OS99 program together with IMAC L. For OS 1 you have to use this special batch file, if OS99 is required. For the remaining OSs please use the standard batch file without start of 'OS99'.

The OS99 program bears this name due to historical reasons. Nevertheless this is an irritating name, because it actually is a PLC program (and not an OS) that is being executed. This PLC program has been generated using the usual tools for any other PLC in IMAC L. It holds the AbvSPU code to process the required PVs for SIC.

#### **Important Hint:**

If you will update a system to a newer AlphaVision version that has a new version of the AbvSPU signal processing with it, you will have to update the AbvSPU of the 'OS 99' program as well. Instructions on this – see below.

#### **Important Hint:**

If the 'OS 99' program is not running, some functions of IMAC L will not work. This includes all SIC related functions. Additionally, the EVENTLOG will not show the user login / logout events (the required data objects are stored in OS 99). Voyage Data in GD 'X2' cannot be entered. If no user rights / SIC / Voyage data are used in a project, this may be acceptable.



**11.5.2. ReCompile the SoftPLC Program for 'OS 99'**

As a rule, the SoftPLC7 program for OS 99 has to be re-compiled once directly on the target hardware of the OS. The reason for this is, that it has been observed in the past, that SoftPLC7 instances being compiled on one PC may not run correctly on a different PC hardware, if only the compiled files are transferred. Compiling directly on the target PC (OS) will avoid this problem.

To re-compile the SoftPLC7 'OS 99' program proceed as described in section '10.8 Converting SoftPLC7 Programs'. In this procedure enter the following directories in the 'Simatic Import' :

- 'Source Folder' : 'C:\vision\SoftPLC7Data\IMACL\_OS99\S7\_Source\')
- 'Destination folder' : 'C:\vision\SoftPLC7Data\IMACL\_OS99\')

**11.5.3. Update the AbvSPU of the SoftPLC Program for 'OS 99'**

If you receive a new version of the AbvSPU signal processing unit, all PLCs will have to receive this new version. This includes the 'OS 99' PLC program as well, because from the software point of view this is a PLC like any other in IMAC L.

To upgrade the AbvSPU of the SoftPLC7 'OS 99' program, proceed as follows:

- Open the Simatic Manager on your Engineering PC
- Click on 'File → Open'
- Point to 'C:\vision\SoftPlc7Data\IMACL\_OS99\S7\_Source\IL\_OS99'
- Click on 'OK' to open
- Perform all steps required to update an IMAC L PLC to a new version of the AbvSPU (see document 'IMAC L Installation Procedure for OS / Engineering PC' in Section 11.5). The OS 99 program has to be handled in precisely the same manner as all other PLCs. The only exception is that the modified program cannot be loaded into the PLC.
- Close the Simatic Manager after successfully compiling the new AbvSPU
- Copy and transfer the directory 'C:\vision\SoftPlc7Data\IMACL\_OS99\S7\_Source' with all its sub-folders from your Engineering PC to your OS1 (e.g. on a USB Stick). Before doing so, delete this folder on OS1 before copying the new version to OS1.
- Re-Compile the SoftPlc7-Program on OS1 (see → 11.5.2. ReCompile the SoftPLC Program for 'OS 99' on page 201).

#### 11.5.4. Engineering for SIC

IMAC L has 16 SIC tokens, which are used to control the access to the 16 technology areas. The following section '[SIC]' from the 'vision.ini' file gives an overview of the function numbers used with SIC in IMAC L:

[SIC]

```
SIC.Entry01=1, 21, 81  
SIC.Entry02=2, 22, 82  
SIC.Entry03=3, 23, 83  
SIC.Entry04=4, 24, 84  
SIC.Entry05=5, 25, 85  
SIC.Entry06=6, 26, 86  
SIC.Entry07=7, 27, 87  
SIC.Entry08=8, 28, 88  
SIC.Entry09=9, 29, 89  
SIC.Entry10=10, 30, 90  
SIC.Entry11=11, 31, 91  
SIC.Entry12=12, 32, 92  
SIC.Entry13=13, 33, 93  
SIC.Entry14=14, 34, 94  
SIC.Entry15=15, 35, 95  
SIC.Entry16=16, 36, 96
```

The first number referenced in each entry is called the '**SIC Number**'. The SIC Number in IMAC L is equal to the technology areas of the system (Range 1 ... 16, not to be modified).

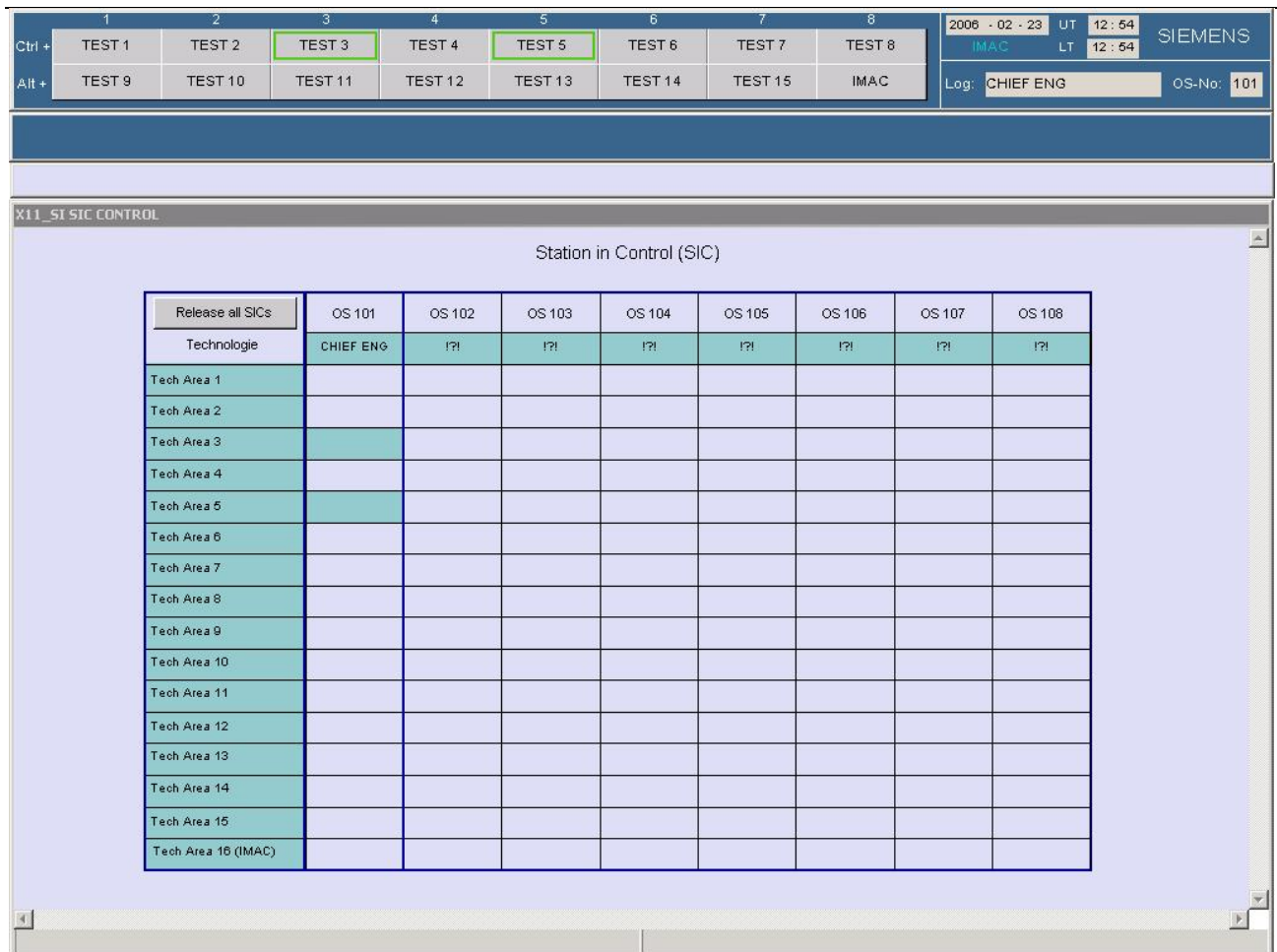
The second number referenced in each entry is called the '**SIC Profile**' (Range 21 ... 36, not to be modified). This SIC Profile Number is automatically added to the respective user profile, if a user receives this SIC (i.e. his rights are temporarily expanded by this function number). The SIC Profile will be automatically removed again from the user profile, if the user releases / loses this SIC.

The third number referenced in each entry is called the '**User Function Number**' (Range 81 ... 96, not to be modified). Only if the user has this User Function Number in his user profile, he will be allowed to request this SIC token at all. Otherwise the request for this SIC will be rejected.

No modification in the 'vision.ini' file is required for SIC.

To interlock the access to MePos and Controls (buttons etc.), the respective 'SIC Profile' number has to be assigned to the respective objects (e.g. 'FNoChange' of a MePo) as the function number for access control.

SIC will then work in a way that a user has to be logged in. Then he can request SIC permissions in the GD 'X11\_SI' (SIC Control) by clicking into the rectangle of the respective technology area of his OS:



The 'X11\_SI' GD will as well show, which SICs are assigned to which OS and which users(s) are logged in on the OSs. The header area will indicate, which SICs a user currently has on his OS, by showing green rectangles in the buttons for the respective technology areas in the header.

The user will only be allowed to request SICs, for which he has the respective 'User Function Number' in his user profile. All other SICs cannot be requested.

Only after having received the SIC, the user will be allowed to control the objects in question (i.e. which are interlocked by the respective 'SIC Profile' number as a function number).

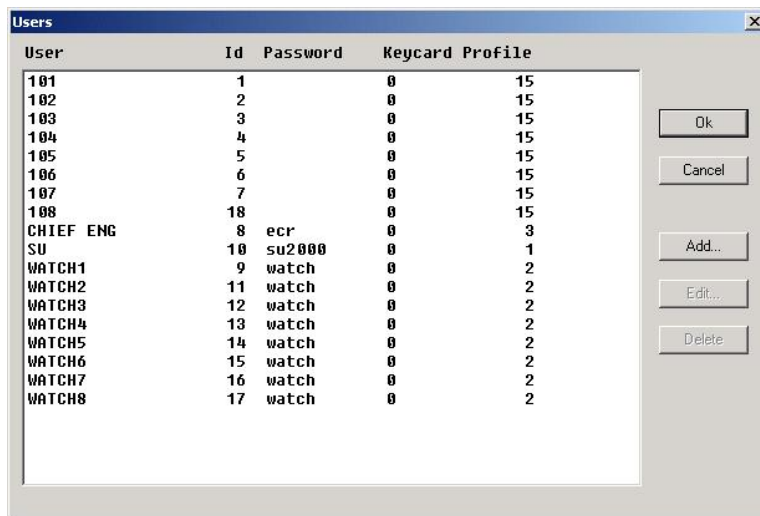
If a SIC is requested, that is currently assigned to a different user on another OS, this SIC will be implicitly taken away from the previous user and be assigned to the user who has requested it.

The resulting basic function of SIC is, that each of the 16 technology areas can only be controlled by one operator / one OS at a time. The operator requires sufficient rights to do so. There is no strict interlock on requesting a SIC, if it should be possessed by another user. If the requesting user has sufficient rights for this SIC, he can simply take it away if he deems this necessary. Thus there can be no interlocks resulting from a user not granting access to a SIC he currently possesses.

SICs will be automatically released, if the user logs out. SICs can be manually released (without a log out) by clicking on the button 'Release all SICs' in GD 'X11\_SIC'.

### 11.5.5. Engineering for Users

User Profiles for IMAC L are modified in 'PMU' → 'AbvBrowser' → 'Users...':



User	Id	Password	Keycard	Profile
101	1		0	15
102	2		0	15
103	3		0	15
104	4		0	15
105	5		0	15
106	6		0	15
107	7		0	15
108	8		0	15
CHIEF ENG	8	ecr	0	3
SU	10	su2000	0	1
WATCH1	9	watch	0	2
WATCH2	11	watch	0	2
WATCH3	12	watch	0	2
WATCH4	13	watch	0	2
WATCH5	14	watch	0	2
WATCH6	15	watch	0	2
WATCH7	16	watch	0	2
WATCH8	17	watch	0	2

For IMAC L please do not delete / add users to this list. This is technically possible, but there are several other settings necessary, for which there is no description on how to do this.

The Users 101 ... 108 are default profiles which are used, if no user is logged in on an OS. These profiles may not have a password, because they have to work all the time. The Profile #15 is mandatory for these users. Do not modify any of these predefined '10x' users at all !

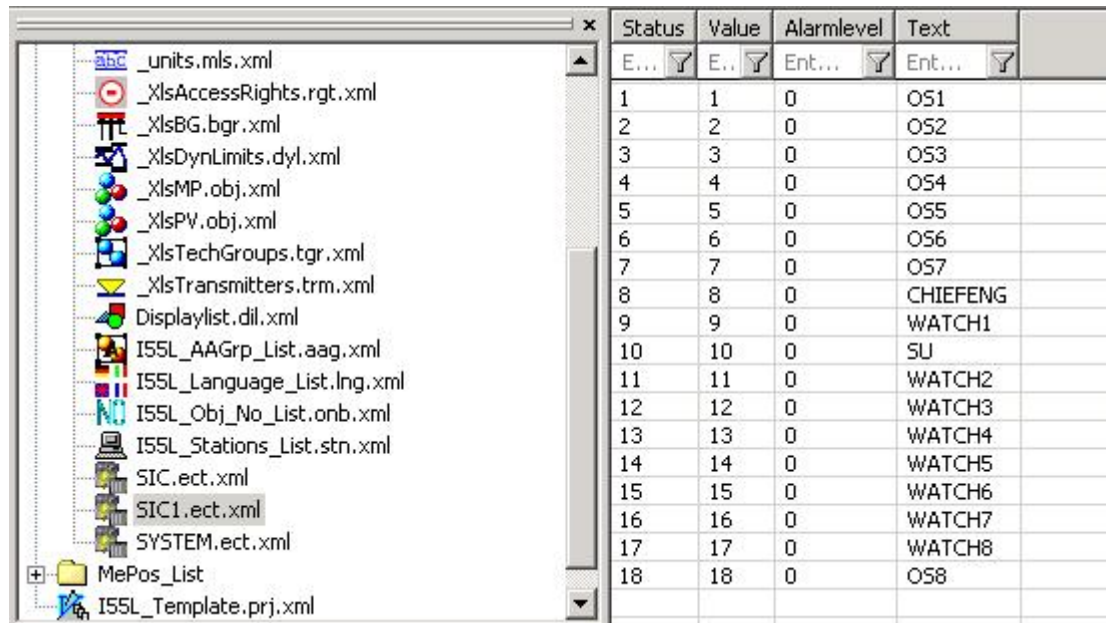
The 'WATCHx' profiles are meant for watch keepers. In the default they all have the profile #2 assigned to them. It is no problem to create copies of profile #2 (see next section) and assign different profiles to different watch keepers, if required.

The 'CHIEF ENG' profile bears an extended set of rights with profile #3. Do not modify the assignment to profile #3, as some functions (e.g. FADEIN / FADEOUT) will require this extended set of user rights, which usually only the 'CHIEF ENG' has.

The 'SU' profile (Super User) bears a complete set of rights with profile #1. Do not modify / rename / change password / change profile for this user, as this is a login used by Siemens Service.

It is permissible to modify the passwords for all users except 'SU'.

It is possible to rename a user in the AbvBrowser. For this you will have to rename it in the users list (edit). Additionally you will have to modify the list in 'AvEdit' → 'Imported' → 'SIC1.ect.xml' accordingly:



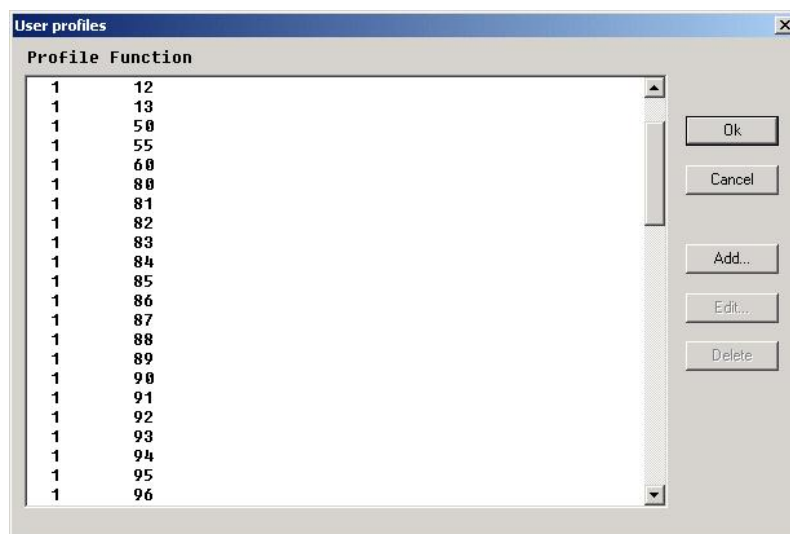
Status	Value	Alarmlevel	Text
E...	E...	Ent...	Ent...
1	1	0	OS1
2	2	0	OS2
3	3	0	OS3
4	4	0	OS4
5	5	0	OS5
6	6	0	OS6
7	7	0	OS7
8	8	0	CHIEFENG
9	9	0	WATCH1
10	10	0	SU
11	11	0	WATCH2
12	12	0	WATCH3
13	13	0	WATCH4
14	14	0	WATCH5
15	15	0	WATCH6
16	16	0	WATCH7
17	17	0	WATCH8
18	18	0	OS8

The texts in the 'SIC1.ect.xml' table are used to enter the login / logout of users in the EVENTLOG. The texts in this list have to fit the user names in the AbvBrowser. Otherwise the entries in the EVENTLOG will be erroneous.

Do not modify any other entries except the 'Text' property in the 'SIC1.ect.xml' file. Do not add or remove entries from this list.

### 11.5.6. Engineering for User Profiles

User Profiles for IMAC L are modified in 'PMU' → 'AbvBrowser' → 'User Profiles...':



Profile	Function
1	12
1	13
1	50
1	55
1	60
1	80
1	81
1	82
1	83
1	84
1	85
1	86
1	87
1	88
1	89
1	90
1	91
1	92
1	93
1	94
1	95
1	96

This user profile list is a plain list, which shows, which profile has which user rights (i.e. function numbers) assigned to it. Rights can be added, edited and deleted with this dialogue.

For IMAC L only the assignments to 'Function' numbers 81 ... 96 have to be modified for each user profile to determine, if the profile grants access to the technology area SICs 1 ... 16. In the screen-shot example, the user profile #1 will have access to all 16 technology area SICs.

Add or delete these entries with function numbers between '81' and '96' as required. Leave all other Function number assignments unmodified ! They are part of the IMAC L system and may be not be modified.

You may create copies of profile #2 (copy all entries one by one) to create own project specific user profiles. Nevertheless even in these copies you may only modify the entries with function numbers between '81' and '96' and leave all other predefined assignments as they were in profile #2.

Do not modify any of the entries for the profiles #1 (Super User), #3 (Chief Engineer) and #15 (default OS rights) without a very good reason. This may jam basic functions of IMAC L if done improperly.

#### 11.5.6.1. IMAC L Function Numbers

Function numbers in IMAC L are predefined and may not be modified. The following list contains some typical function numbers with their use in IMAC L :

Function Number	Description
1	Set Parameters (e.g. Limits) for Measuring Points
3	Set System Parameters (e.g. Time & Date, Printer Queue Name)
12	Fade In / Fade Out of Measuring Points
98	Set Voyage Data / Set Event Printing On / Off
99	Change User Password

### 11.6. Terminating the IMAC L tasks of an OS

It is not allowed to terminate the IMAC L tasks running on an OS using the Windows task manager. This can potentially leave parts of the IMAC L tasks still loaded in the system (although invisible for the task manager). If the task manager has been used to kill any of the IMAC L tasks, it is necessary to reboot the OS to make IMAC L run again.

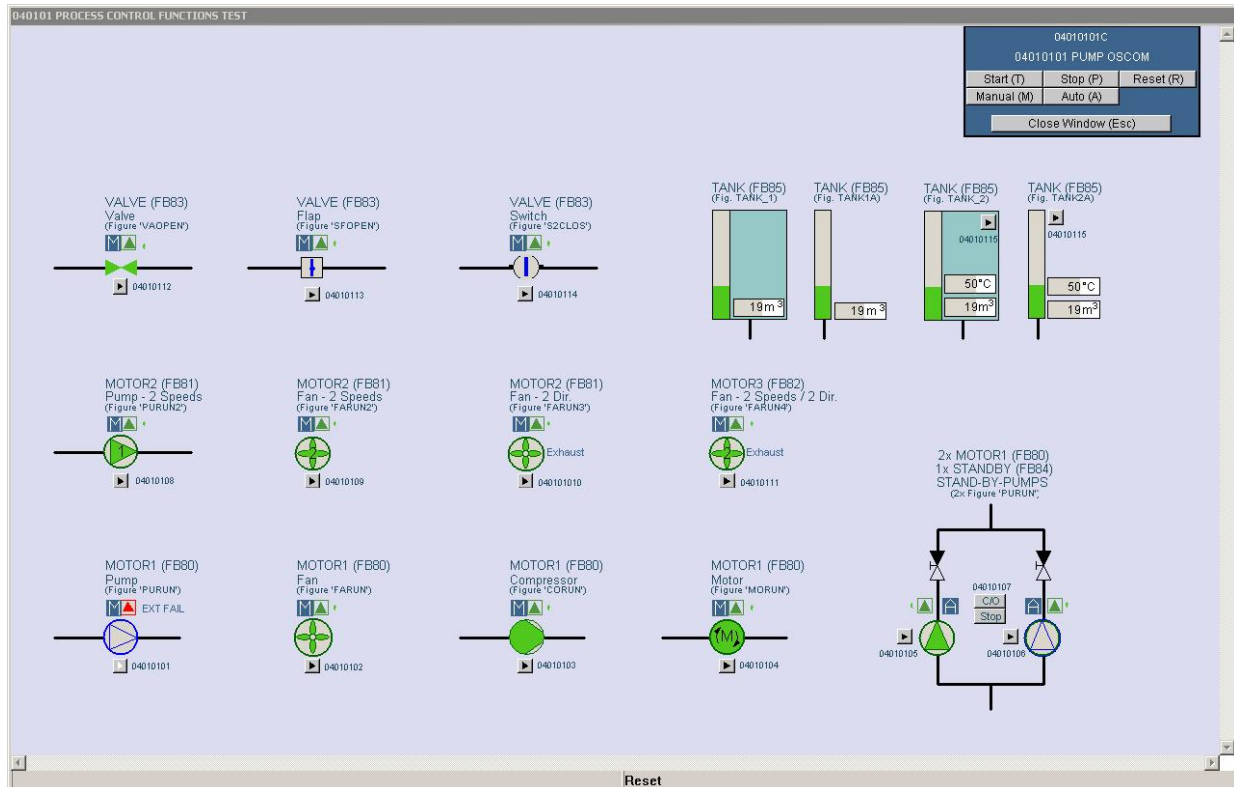
The usual procedure for terminating tasks is:

- **Udp2AV.exe:** click on the 'Udp2Av' window to make it the active window and then repeatedly press 'ESC'. The task will take some time to terminate and may sometimes throw an error message window while terminating. This is normal and can be ignored.
- **SoftPlc7.exe:** click on the 'Close' box of the task. The task will terminate immediately.
- **Prcs.exe:** Press 'Ctrl-X' (or any other 'Backdoor Exit' you have) to call the 'Usr\_Exit()' job. If you have no backdoor, you will have to use the task manager to kill this task. Then it is absolutely necessary to reboot the PC before making it an OS again !



## 11.7. Dynamic HMI Elements for Process Control

The template files in the IMAC L standard distribution contain HMI representations of the Function Blocks described in section → 10.1 Introduction to Process Control Functions on page 134 ff. This includes a template process GD named '040101' with a number of process objects as they are used on board of ships. The name of this GD suggests, that all Objects including their MePos are assigned to Technology Area '04' and to Section '0401' within this Technology Area.



The template files contain all required MePos, parameterizations and function calls to make these objects work. It is recommended to take these templates as a starting point for engineering. It is not recommended to start the HMI graphics pictures design from scratch.

The following descriptions refer to these templates as an example for naming of objects. Please remember, that other GD names / other object names will require appropriate renaming of these objects while 'copy / pasting' them. Remember to set the 'Technology Area' and 'Section' of each MePo correctly to fit your requirements.

As depicted in the screenshot above, the IMAC L control function blocks can have different HMI representations – depending on their use on board:

- FB80 - MOTOR1: Pump, Fan, Compressor, Motor, Standby-Pump
- FB81 - MOTOR2: Pump 2 Speeds, Fan 2 Speeds, Fan 2 Directions
- FB82 . MOTOR3: Fan 2 Speeds & 2 Directions
- FB83 - VALVE: Valve, Flap, Switch
- FB84 - STANDBY: Standby Control for 2 Pumps
- FB85 - TANK: Tank

These Representations are predefined and ready for use. Their use and application is described in the sections below. Other representations are possible, but their design requires additional special engineering, which is not covered in this engineering manual.

### 11.7.1. MePos required for HMI access to 'Control Function' Objects

Each instance of a 'Control Function' object requires a set of MePos, which is required to connect this object to the IMAC L HMI. Without these MePos neither alarming nor the HMI monitoring and control access from a GD are possible.

The MePos as described in the list below in this section are required for each single instance of a process object.

The MePos for each object have to follow a strict naming scheme, where only the basic name of the object itself is freely selectable. This basic name has to be identical for all MePos belonging to the same Process Object. In the list below this basic name is referred to as 'xxxxxxx'. This basic name has to be unique for each single Process Object within a whole IMAC L project.

Index letters are being used to distinguish the single MePos belonging to the same Process Object. The assignment of these letters is fixed according to the scheme as shown in the list below in this section (highlighted in red font).

In the list below 'yyy' is assumed to be the instance number of the respective Control Function (MOTOR1, MOTOR2, MOTOR3, VALVE, STANDBY, TANK) in each PCU. 'yyy' starts with '001' for the first control function of this type in each PCU and may not have gaps (continuous numbering).

All Data Sources for the predefined number of objects do exist in the template files DB38x and are correctly referenced to in the Function Calls (FB18x) for the Process Objects.

Control Function	MePos Names	Type of MePo	Data Source ('Address') (see DB symbols to identify cell)	Comment
FB 80 'MOTOR1'	xxxxxxx xxxxxxxA xxxxxxxC xxxxxxxS	MP Analogue, 9 States MP Binary with Alarm PV Numerical UINT PV Numerical UINT	DB380.MOTOR1_yyy_OSALARM DB380.MOTOR1_yyy_ASTAT.12 DB380.MOTOR1_yyy_OSCOM DB380.MOTOR1_yyy_OSSTAT	Alarm Word Status Word.Alarm OS Cmd Word OS Status Word
FB 81 'MOTOR2'	xxxxxxx xxxxxxxA xxxxxxxC xxxxxxxS	MP Analogue, 9 States MP Binary with Alarm PV Numerical UINT PV Numerical UINT	DB381.MOTOR1_yyy_OSALARM DB381.MOTOR1_yyy_ASTAT.12 DB381.MOTOR1_yyy_OSCOM DB381.MOTOR1_yyy_OSSTAT	Alarm Word Status Word.Alarm OS Cmd Word OS Status Word
FB 82 'MOTOR3'	xxxxxxx xxxxxxxA xxxxxxxC xxxxxxxS	MP Analogue, 9 States MP Binary with Alarm PV Numerical UINT PV Numerical UINT	DB382.MOTOR1_yyy_OSALARM DB382.MOTOR1_yyy_ASTAT.12 DB382.MOTOR1_yyy_OSCOM DB382.MOTOR1_yyy_OSSTAT	Alarm Word Status Word.Alarm OS Cmd Word OS Status Word
FB 83 'VALVE'	xxxxxxx xxxxxxxA xxxxxxxC xxxxxxxS	MP Analogue, 9 States MP Binary with Alarm PV Numerical UINT PV Numerical UINT	DB383.VALVE_yyy_OSALARM DB383.VALVE_yyy_ASTAT.15 DB383.VALVE_yyy_OSCOM DB383.VALVE_yyy_OSSTAT	Alarm Word Status Word.Alarm OS Cmd Word OS Status Word
FB 84 'STANDBY'	xxxxxxxC	PV Numerical UINT	DB384.STANDBY_yyy_OSCOM	OS Cmd Word
FB 85 'TANK'	xxxxxxx xxxxxxxT xxxxxxxD xxxxxxxR xxxxxxxX xxxxxxxL xxxxxxxM xxxxxxxV xxxxxxxS	Standard Analogue MP Standard Analogue MP PV Numerical REAL PV Numerical REAL PV Numerical REAL Analogue MP int. REAL Analogue MP int. REAL Analogue MP int. REAL Dummy MePo for ScaleGrp	(Level or Pressure Analogue Input) (Temperature Analogue Input) DB385.TANK_yyy_XDensity DB385.TANK_yyy_XRefTemp DB385.TANK_yyy_XVEC DB385.TANK_yyy_YLevel DB385.TANK_yyy_YMass DB385.TANK_yyy_YVolume e.g. Level or Pressure Analogue Input	(e.g. PEW xxx) (e.g. PEW xxx) Density from OS Ref.-Temp from OS Exp.Coeff. from OS Computed Level Computed Mass Computed Volume To create ScaleGrp

If setting up these MePos for the first time, it is strongly recommended to 'copy / paste' them from the Excel MePo list delivered with the template files:

- Select the rows in the template Excel MePo list containing the required MePos for one object
- Click 'Edit' → 'Copy'
- Select 4 empty rows at the end of the target MePo List.
- Click 'Edit' → 'Paste Special' → 'Values'

Usually after 'Copy / Paste of a MePo set for such a Process Object, the following fields have to be checked / modified for each MePo:

- PointID (see naming Scheme above)
- Description of Point (according to technological use on board)
- Station (PCU, where the Control Function is allocated)
- Section Data (Section to which the Control Function belongs)
- TechID (Technology Area, to which the Control Function belongs)
- Address (point to the Data Source according to list above). Observe, that the Simatic Symbols may not be used in the MePo Frontend !

### 11.7.2. System Objects required for each GD

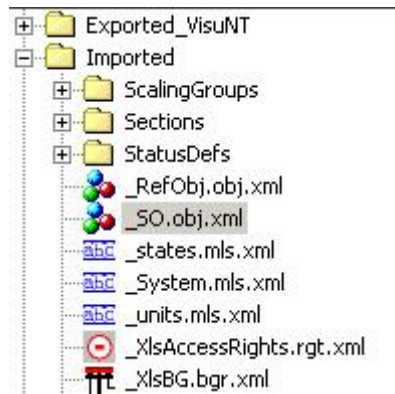
Each Graphics Display (GD) in IMAC L requires a set of two individual SYSTEM OBJECTS for exclusive use in this specific GD. These objects are required for the Operation of each GD and have to be defined in IMAC L before the setting of dynamic attributes for this GD can begin.

The naming scheme used here assumes, that 'zzzzzz' is that name of the GD for which the SYSTEM OBJECTS are being created. The following two SYSTEM OBJECTS have to be created:



Object Name	Object Type	Target in AvEdit	Purpose
<b>XSzzzzzzz</b>	System Object	_SO.obj.xml	Switch Control Window Figure
<b>XSzzzzzzzA</b>	System Object	_SO.obj.xml	Make correct ROUTING BUTTON blink

To define or modify a SYSTEM OBJECT in AvEdit, proceed as follows :

- Double-Click on the 'Imported' Folder
- Double-Click on the '\_SO.obj.xml' File



- In the middle window a list of all previously defined SYSTEM OBJECTS will be displayed.
- To modify an existing SYSTEM OBJECT, just click on the appropriate line in the middle window. Edit as required.
- To add a new SYSTEM OBJECT, click on 'New Object' (🌐) in the Menu Bar. A new SYSTEM OBJECT will appear with a default name.
- Then edit the SYSTEM OBJECT name and description in the right window. All other parameters have to stay as they are by default (see screenshot below).

- Click on the 'Apply' button (  ) in the right window to save the changes to this list.
- Click on the 'Save' button (  ) of AvET.
- The new SYSTEM OBJECT is now ready for use :

### 11.7.3. Reference Objects required for each GD

Each Graphics Display (GD) in IMAC L requires a set of five individual REFERENCE OBJECTS for exclusive use in this specific GD. These objects are required for the Operation of each GD and have to be defined in IMAC L before the setting of dynamic attributes for this GD can begin.

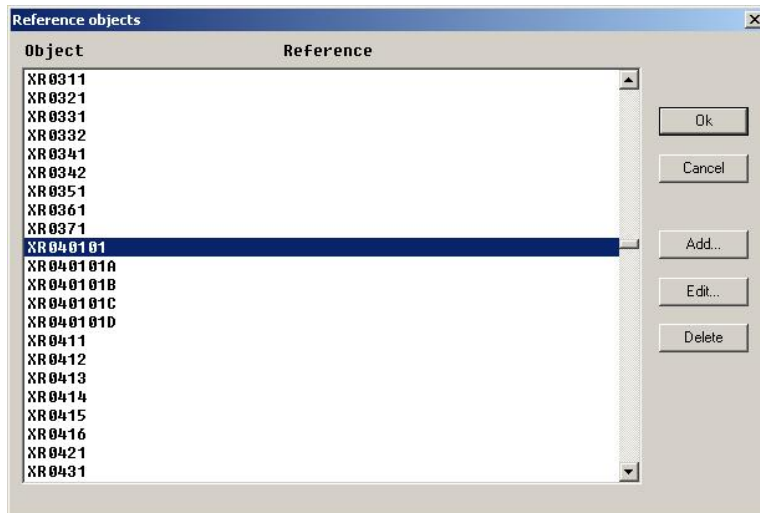
The naming scheme used here assumes, that 'zzzzzz' is that name of the GD for which the REFERENCE OBJECTS are being created. 'zzzzzz' may have between 2 and 6 letters in length and must be unique for the respective GD. The following five REFERENCE OBJECTS have to be created:

Object Name	Object Type	Purpose
<b>XRzzzzzzz</b>	Reference Object	Set PointID, Name and target object for control commands in the Control Window
<b>XRzzzzzzzA</b>	Reference Object	Rference to 'Input MePo' #1 from Control Window'
<b>XRzzzzzzzB</b>	Reference Object	Rference to 'Input MePo' #2 from Control Window'
<b>XRzzzzzzzC</b>	Reference Object	Rference to 'Input MePo' #3 from Control Window'
<b>XRzzzzzzzD</b>	Reference Object	Rference to 'Input MePo' #4 from Control Window'

REFERENCE OBJECTS for IMAC L are modified in 'PMU' → 'AbvBrowser' → 'Reference Objects...'.  
To define or modify a REFERENCE OBJECT, proceed as follows :

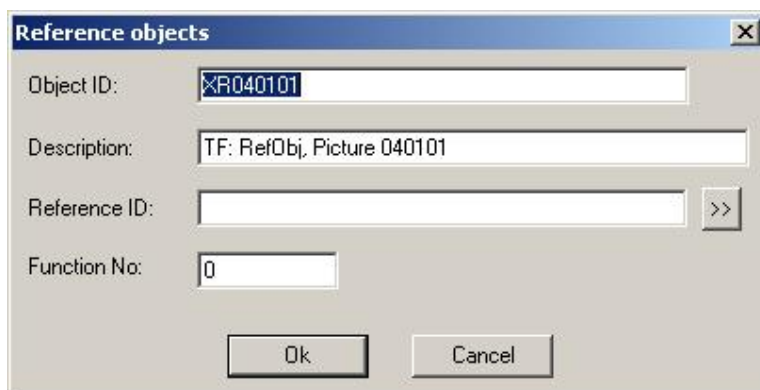
- Double-Click on the 'PMU' icon on your desktop
- Click on the 'Start' → 'AbvBrowser'
- Click on 'File' → 'Reference Objects...'

The following Window will Open:



To add a new REFERENCE OBJECT, proceed as follows :

- You may optionally select an existing REFERENCE OBJECT (from which the settings will be copied, e.g. XR040101 from IMAC L template files). If you do not select an existing object, the next window will be opened with empty entry fields for the new REFERENCE OBJECT.
- Click on the 'Add...' button
- A new Window will open, showing the properties of the new REFERENCE OBJECT:



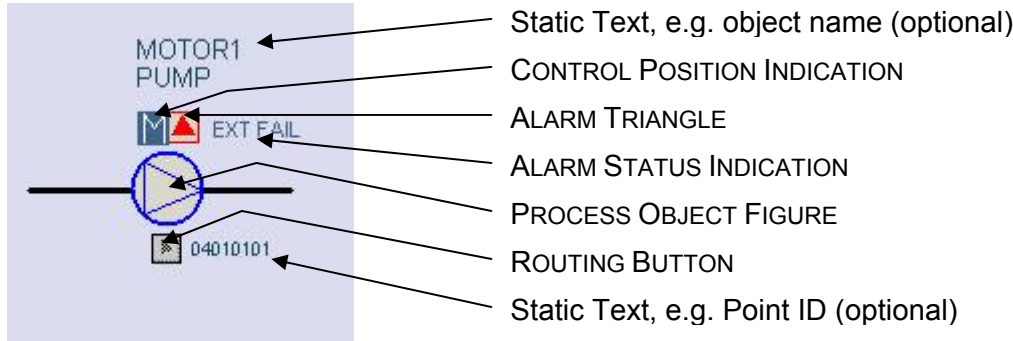
- Edit the 'Object ID' (name) and 'Description' fields to fit the new REFERENCE OBJECT
- Click on 'OK'

To edit an existing REFERENCE OBJECT, proceed as follows :

- Select the existing REFERENCE OBJECT that has to be edited. To easily select the desired point without scrolling, you may simply start to type the desired objects name (but rather fast, as too long gaps start a new entry). The object will be automatically selected in the list.
- Click on the 'Edit...' button
- Modify the properties as required
- Click on 'OK'

### 11.7.4. Dynamic Attributes for 'Indication and Control' HMI Objects

An 'Indication and Control' object in IMAC L consists of several individual components according to the next screen shot:



All of these components have to be fitted with the correct dynamic attributes. This section describes, how this setting of dynamic attributes works.

#### 11.7.4.1. Starting a new Graphics display

It is good practice not to start a graphics display from scratch, but to use one of the many existing templates instead. A Good starting point is the GD 040101 from the IMAC L distribution.

To create a copy of this GD:

- Double-Click on the 'PMU' icon on your desktop
- Select the IMAC L template project ('IMAC\_L') and version (1.00).
- Click on 'Start' → 'AbvEditor'
- Click on 'File' → 'Open'
- Select 'Application Area' from the 'Type' combo box in the lower part of the window.
- Select the Display '040101'
- Click on 'OK'
- Click on 'File' → 'Save As'
- Enter a new GD name and description
- Click on 'OK'

Another possibility is to transfer an existing picture from one project to another project. To achieve this, the picture has to be 'exported' from the source project and 'imported' to the target project.

To export a picture from a 'source' project:

- Double-Click on the 'PMU' icon on your desktop
- Click on the source 'project' and 'version'
- Click on 'Version' → 'Export' → 'Picture...'
- A new window opens. Select the desired picture in the upper part of the window.
- Enter the name of a target directory (has to exist) in the entry field in the lower part of the window. The exported files will be placed in this directory.
- Click on 'OK'



To Import a picture to a 'target' project:

- Double-Click on the 'PMU' icon on your desktop
- Click on the target 'project' and 'version'
- Click on 'Version' → 'Import' → 'Picture...'
- A new window opens.
- Enter the name of the directory (has to exist) in the entry field in the upper part of the window. The imported files will be read from this directory.
- Select the desired picture in the lower part of the window (list shows all objects found in this directory).
- Click on 'OK'

The same procedure (export / import) works for other IMAC L HMI objects (e.g. figures, condition tables) in the same way.

If you use such a template from another project, please make sure to use only the static part of such a picture. You will have to delete all dynamic objects from this picture and replace them with the templates from the IMAC L distribution, GD 040101. This can be easily done by 'copy & paste' from this GD within your project.

All instructions in the following sections assume, that you have opened this (your) GD in the 'PMU' → 'AbvEditor' application.

#### 11.7.4.2. Editing properties of static text fields

To Edit the text in a static text field:

- Double-Click on the static text field
- The Text field opens for editing:



- Edit the text as required
- Press 'Enter' to finish

If the size of a text field does not match the required size, the text field can be resized without re-sizing the text / font:

- Single-Click on the static text field
- The Text field shows its handles at its corners:



- Resize the text field using these handles

The properties of a text field (e.g. alignment, font, colour) can be edited as follows:

- Single-Click on the static text field
- Click on 'Edit' → 'Attributes Sequential'
- Modify attributes in the lower part of the dialogue
- Click on 'Close'

### 11.7.4.3. Setting dynamic Attributes for Routing Buttons

ROUTING BUTTONS are used in IMAC L to select a PROCESS OBJECT (e.g. a pump) for control operation.



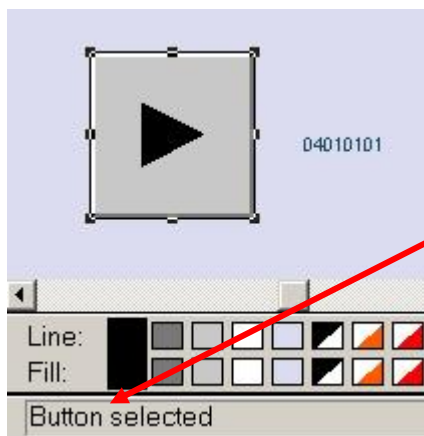
If the operator clicks on this ROUTING BUTTON, the triangle in the ROUTING BUTTON will start to flash to indicate that this object is now selected for control. In the same moment the CONTROL WINDOW in the upper right corner of the GD will appear (or change its appearance if it was already open). The CONTROL WINDOW will show the Point ID, Description Text and control elements of this specific PROCESS OBJECT.

All ROUTING BUTTONS in one GD require a logical numbering, starting from '1' for the first ROUTING BUTTON. To set the dynamic attributes of a ROUTING BUTTON, it is necessary to know this number 'n'. In the following example, the screenshots will show the ROUTING BUTTON #1 to be edited.

Tanks using the figures 'TANK\_1' or 'TANK1A' do not need a control window and do not have a ROUTING BUTTON.

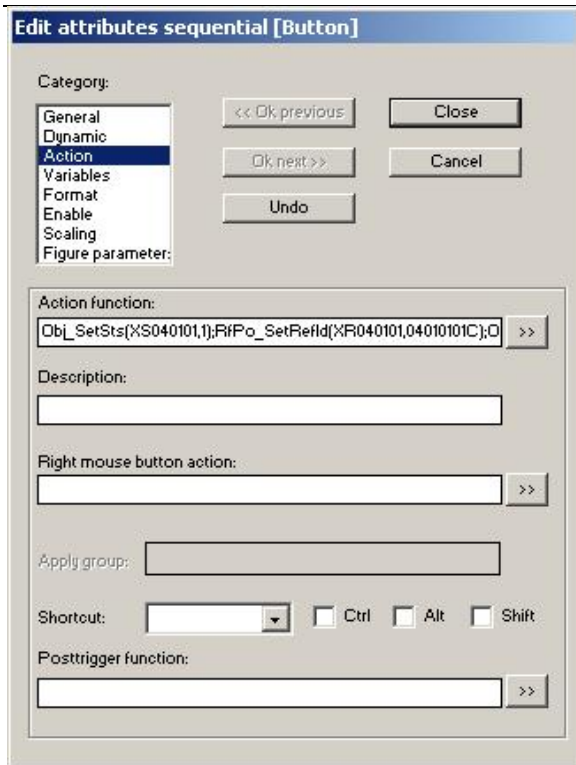
The first step is setting of the 'Action' to be performed, if the operator clicks on this button:

- Single-Click on the ROUTING BUTTON
- Handles will appear around the button.
- The Editor shows 'Button selected' in its lower left corner (observe this text indication to see, if you have selected the right object):

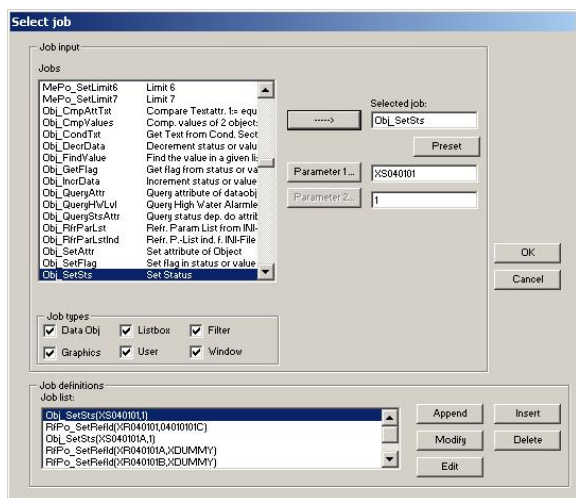


Indication of selected  
type of object(s)

- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Action'
- The following dialogue appears:

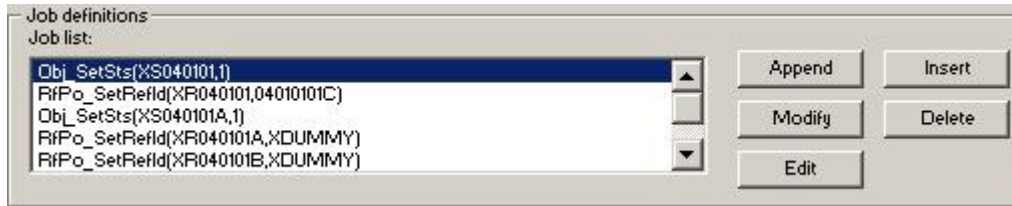


- Click on the '>>' button right of the 'Action Function' field.
- The following dialogue appears:



- The lower part of this dialogue contains a job list containing 7 jobs. This job list has to be modified according to the table below in this section.
- To modify one of the jobs, double click on it. Then modify the job parameters in the upper right corner of the dialogue to fit the requirements of the list below and click on 'Modify' to finish with this job.
- Modify all other jobs as necessary.
- Click on 'OK' if finished with all 7 jobs.
- Click on 'Close' to finish editing of the 'Action' of the ROUTING BUTTON.

The jobs list in the 'Job List' window looks like this:



The seven jobs in the job list of each job have to be modified as follows:

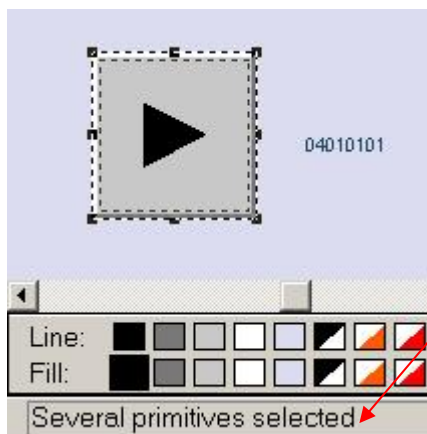
Job No.	'Job'	Parameter 1	Parameter 2	Comment
1	Obj_SetSts	XSzzzzzz	m	Select, which type of control window is required (see list below)
2	RfPo_SetRefID	XRzzzzzz	xxxxxxxxC	Set REFERENCE OBJECT to the PROCESS OBJECTS 'Control MePo', so that the POINTID & Text of the point are correctly displayed in the CONTROL WINDOW and the control input from the button ends up at the right object.
3	Obj_SetSts	XSzzzzzzA	n	Select the correct ROUTING BUTTON to blink. n = number of this button.
4	RfPo_SetRefID	XRzzzzzzA	XDUMMY xxxxxxxxD	No Inputs (for all common objects) Density Input MePo (for TANKS)
5	RfPo_SetRefID	XRzzzzzzB	XDUMMY xxxxxxxxR	No Inputs (for all common objects) Temp. Input MePo(for TANKS)
6	RfPo_SetRefID	XRzzzzzzC	XDUMMY xxxxxxxxX	No Inputs (for all common objects) Volume Expansion Coefficient Input MePo(for TANKS)
7	RfPo_SetRefID	XRzzzzzzD	XDUMMY	No Inputs (for all objects)

Where :

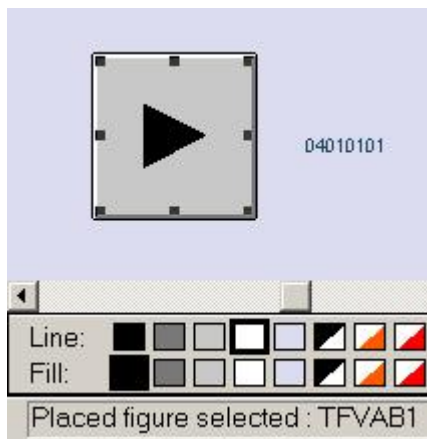
- zzzzzz = Name of the GD
- m = Type key for Process Object
  - 1 = MOTOR1 (Start / Stop)
  - 2 = MOTOR2 – Motor with 2 Speeds (Slow / Fast)
  - 3 = VALVE (Open / Close / Toggle)
  - 4 = TANK
  - 5 = MOTOR3 – Fan with 2 Speeds & 2 Directions
  - 6 = MOTOR2 – Fan with 2 Directions (Exhaust / Supply)
  - 7 = MOTOR2 – Motor with 2 Directions (Ahead / Reverse)
- xxxxxxxx = Point ID of the PROCESS OBJECT, to which this ROUTING BUTTON belongs
- n = Logical number of the ROUTING BUTTON in this GD

The second step for a ROUTING BUTTON is setting of the dynamic Attribute of the triangle symbol on the button (which is blinking, if the button is selected):

- Click on 'Setup' → 'Zoom' → '500%' (Remark: For this action you will have to switch to a high zoom factor (500%) – otherwise the arrangement of the triangle on the button will not work correctly)
- Use the slider-bars to locate the ROUTING BUTTON to be edited
- Single-Click on the ROUTING BUTTON
- Handles will appear around the button.
- Click on 'Arrange' → 'Ungroup'
- The Editor shows the indication 'Several primitives selected' in its lower left corner :

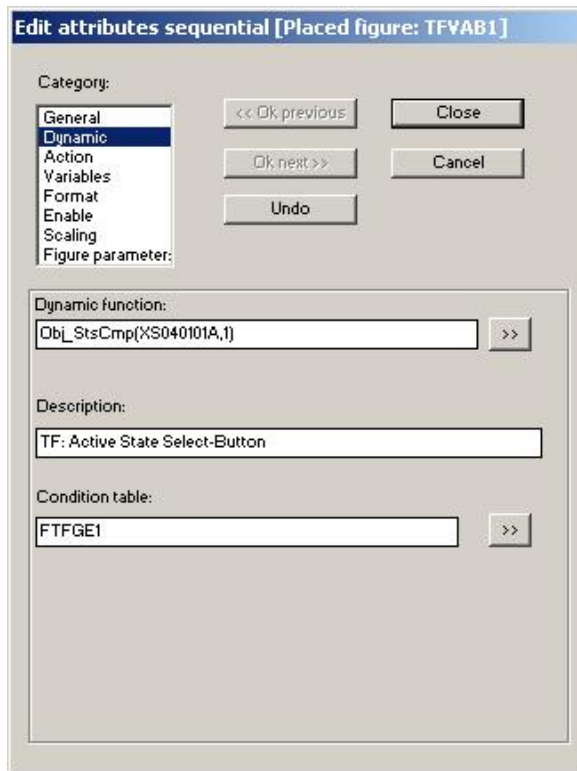


- Single-Click on the ROUTING BUTTON until 'Placed figure selected: TFVAB1' is displayed



- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Dynamic'

- The following dialogue appears:



- Edit the 'Dynamic function' field to be as follows:

Job No.	'Job'	Parameter 1	Parameter 2	Comment
1	Obj_StsCmp	XSzzzzzzA	n	If the system variable XSzzzzzzA is being set to the value 'n', this triangle symbol will blink. If its value is different from 'n' the triangle will be statically black.

Where :

- o **zzzzzz** = Name of the GD
- o **n** = Logical number of the ROUTING BUTTON in this GD

- Click on 'Close' to finish editing of the 'Dynamic' of the ROUTING BUTTON.

The grouping of the ROUTING BUTTON itself and it's triangle symbol has been removed in the process above. It has to be re-established (else the triangle will look garbled and have a wrong size:

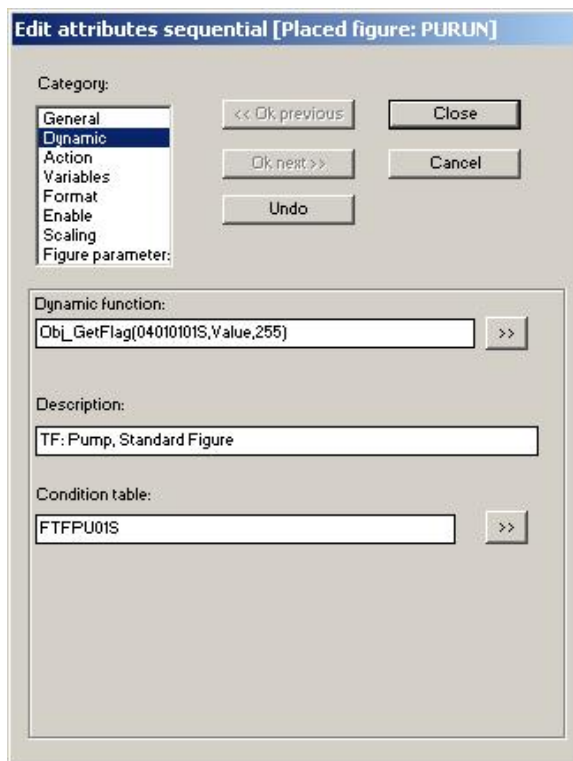
- Use the mouse to catch both the ROUTING BUTTON and the triangle figure using a 'click & drag' motion around these two objects. Be careful to catch no other objects.
  - The Editor again shows the indication 'Several primitives selected' in its lower left corner
  - Click on 'Arrange' → 'Label Button'.
- Remark: It is important, that this step is being performed in the 'Zoom 500 %' setting. Otherwise results may not be as desired.
- Click on 'Setup' → 'Zoom' → 'Fit to Screen'



### 11.7.4.4. Setting dynamic Attributes for Process Object Figures (except TANK, STANDBY)

For an existing Object (e.g. created by 'Copy & Paste' from template GD 040101) :

- Single-Click on the figure to be modified
- Handles will appear around the figure
- The Editor shows 'Figure selected: ' in its lower left corner (The name of the basic figure depends on the object type)
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Dynamic'
- The following dialogue appears:



- Edit the 'Dynamic function' field to be as follows:














Job No.	'Job'	Parameter 1	Parameter 2	Parameter 3	Comment
1	Obj_GetFlag	xxxxxxxxS	Value	255	Status Word (low Byte)

Where :

- o **xxxxxxxx** = Point ID of the Process Object, to which this Figure belongs. **xxxxxxxxS** points to the MePo that delivers the 'OS Status Word' of this object.

Remark: Parameter 3 contains a Bit Mask (255), which addresses only the lower eight bits from the status word.

If a Process Object Figure is created using the 'Insert' → 'Figure' function of the editor, the basic figure being placed in the GD and the condition table being applied to this figure have to stick to a fixed scheme according to the following list:

Control Function.	Use (technological Function)	Figure	Figure Name	Condition Table
MOTOR 1	Pump		PURUN	FTPU01S
	Fan		FARUN	FTFFA01S
	Compressor		CORUN	FTFCO03S
	Motor		MORUN	FTFMOGE2
MOTOR2	Pump with 2 Speeds		PURUN2	FTFPU03S
	Fan with 2 Speeds		FARUN2	FTFFA03S
	Fan with 2 Directions		FARUN3	FTFFA12S
	Direction Indication for Fan with 2 Directions		TFEXHS	FTFADIL2 (left aligned) FTFADIR2 (right aligned)
MOTOR3	Fan with 2 Speeds & 2 Directions		FARUN4	FTFFA05S
	Direction Indication for Fan with 2 Speeds & 2 Directions,		TFEXH2	FTFADIL3 (left aligned) FTFADIR3 (right aligned)
VALVE	Valve		VAOPEN	FTFVA01S
	Flap		SFOPEN	FTFFL02S
	Switch		S2CLOS	FTSW02

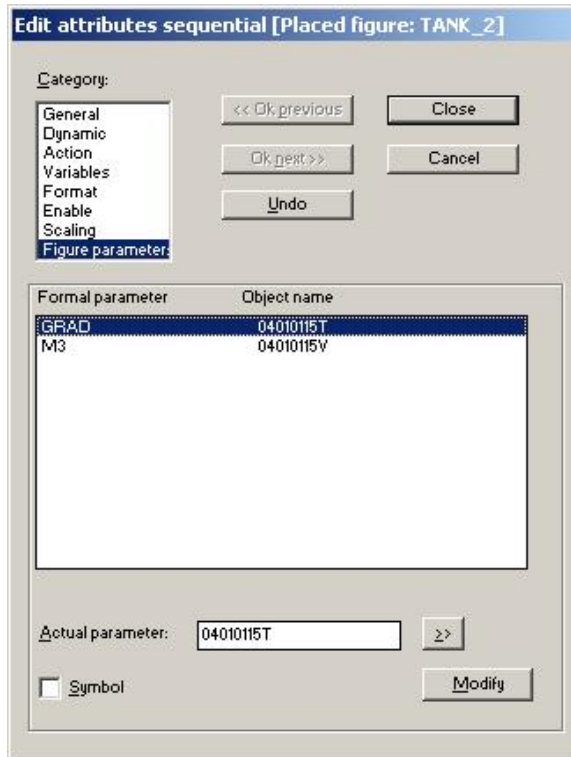
Some figures need to be rotated to fit into the context of a GD. After placing the figure, select this new figure and rotate it using the 'Arrange' → 'Rotate' function of the AbvEditor. By rotating the basic figure, the dynamic representation of the whole dynamic object will be rotated in the HMI.

### **Important Remark:**

There are many Figures and Figure Condition Tables with very similar names and meanings in the IMAC L template files. With Figures and Condition Tables, many ways will lead to Rome, but IMAC L has chosen the one as described above. Please take care to stay strictly with the scheme as shown in the list above. Otherwise serious problems may arise during the setting of dynamic parameters to your GDs.

### 11.7.4.5. Setting dynamic Attributes for Process Object Figure 'Tank'

- Single-Click on the tank figure to be modified
- Handles will appear around the figure
- The Editor shows 'Figure selected: TANK....' in its lower left corner
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Figure Parameters'
- The following dialogue appears:



- Edit the 'Formal Parameter' entries to be as follows (click on 'Modify' after each entry to accept each single edit) :

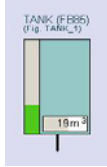
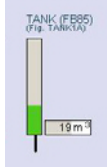
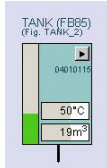
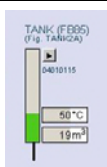
Formal Parameter	Parameter 1	Comment
GRAD	xxxxxxxxT	Tank Media Temperature Indication, only required at figures 'TANK_2' and 'TANK2A'.
M3	xxxxxxxxV	Tank Volume Indication.

Where :

- o xxxxxxxx = Point ID of the Process Object, to which this Figure belongs.

- Click on 'Close'

The object of type 'TANK' (FB85) has four possible representations as a figure:

Control Function.	Use (technological Function)	Figure	Basic Figure Name
TANK	Simple Tank (large figure)		TANK_1
	Simple Tank (small figure)		TANK1A
	Tank with Temperature Measurement (large figure)		TANK_2
	Tank with Temperature Measurement (small figure)		TANK2A

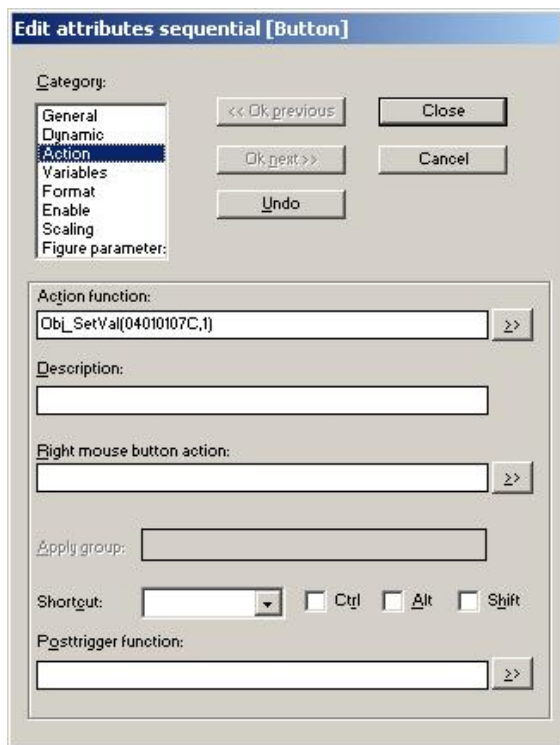
The 'TANK...' figures do not require a condition table. The values indicated in the figure are passed on to the figure using 'Figure Parameters' (see above in this section).

### 11.7.4.6. Setting dynamic Attributes for 'STANDBY' Control Buttons

The IMAC L STANDBY function has no own figure representation. Nevertheless it has a set of two buttons, which will require the setting of dynamic parameters:



- Single-Click on the button to be modified
- Handles will appear around the button
- The Editor shows 'Button selected' in its lower left corner
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Action'
- The following dialogue appears:



- Edit the 'Action Function' entry to be as follows (depending on the button being modified):

Button	Parameter 1	Parameter 2	Comment
C/O-Button (Change Over)	xxxxxxxxxC	1	Set bit '0' of Control Word
Stop-Button	xxxxxxxxxC	2	Set bit '1' of Control Word

Where :

- o xxxxxxxx = Point ID of the Process Object, to which this button belongs – here the Standby function.

- Click on 'Close'
- Repeat these steps for the second button

### 11.7.4.7. Setting dynamic Attributes for the Alarm Triangles

- Single-Click on the ALARM TRIANGLE to be modified
- Handles will appear around the ALARM TRIANGLE
- The Editor shows 'Figure selected : HA' in its lower left corner
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Dynamic'
- The following dialogue appears:

- Edit the 'Dynamic function' field to be as follows:

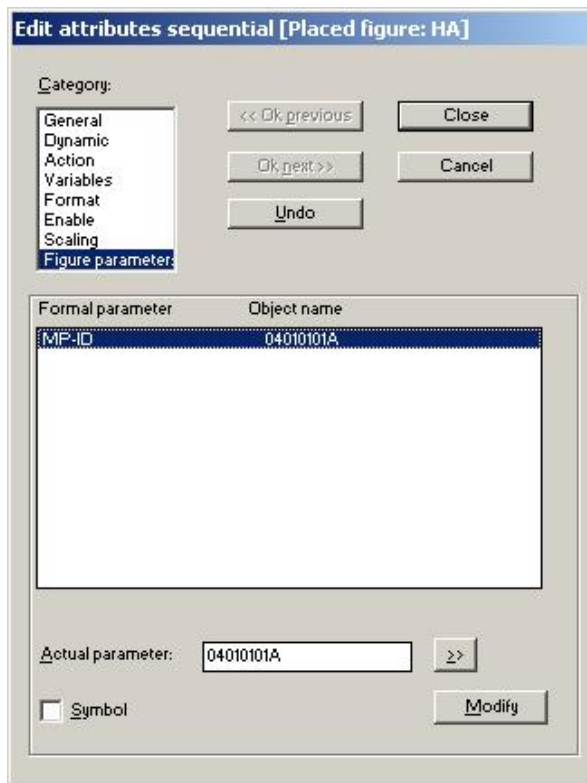
Job No.	'Job'	Parameter 1	Parameter 2	Comment
1	Obj_QueryAttr	xxxxxxxxA	AlarmAct	OS Alarm Bit of object

Where :

- o **xxxxxxxx** = Point ID of the Process Object, to which this Alarm Triangle belongs.



- Click on the category 'Figure Parameter:'
- The following dialogue appears:



- Click on the Formal parameter 'MP-ID' in the 'Formal parameter' window
- Modify the 'Actual Parameter' to be as follows:

Actual Parameter	Comment
xxxxxxxxxA	OS Alarm Word from object, handed over to figure

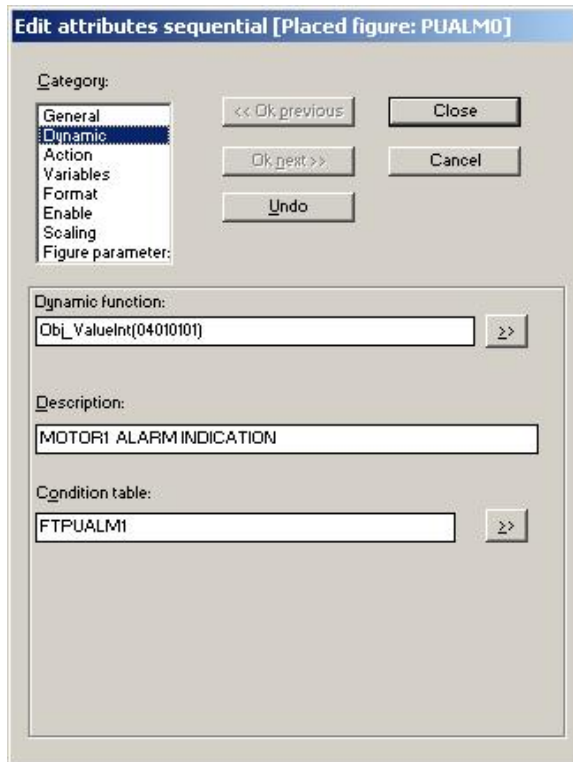
Where :

- o xxxxxxxx = Point ID of the Process Object, to which this Alarm Triangle belongs.

- Click on 'Modify'
- Click on 'Close'

### 11.7.4.8. Setting dynamic Attributes for the Alarm Status Indications

- Single-Click on the ALARM STATUS INDICATION to be modified
- Handles will appear around the ALARM STATUS INDICATION
- The Editor shows 'Figure selected : PUALM0' (or PUALR0) in its lower left corner
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Dynamic'
- The following dialogue appears:



- Edit the 'Dynamic function' field to be as follows:

Job No.	'Job'	Parameter 1	Comment
1	Obj_ValueInt	xxxxxxxx	OS Alarm Word of object

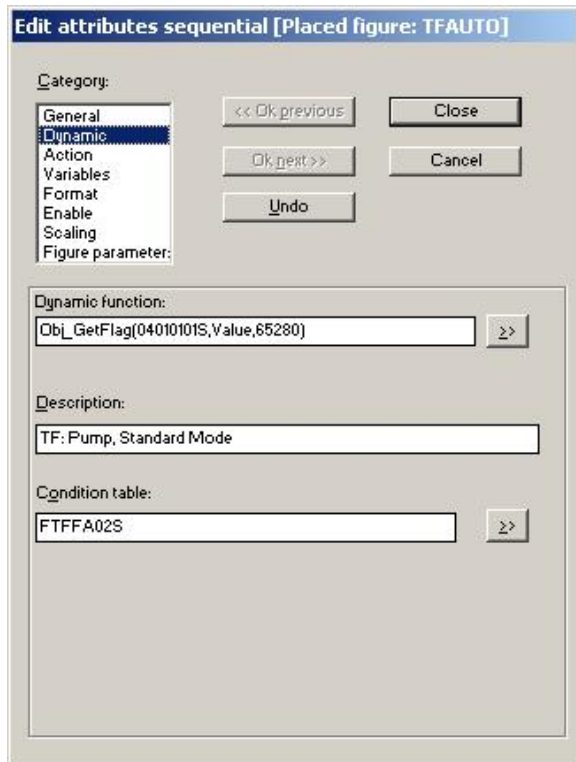
Where :

- o **xxxxxxxx** = Point ID of the Process Object, to which this Alarm Status Indication belongs.

- Edit the 'Condition Table' field to be as follows:
  - For the 'Text left aligned' version, the 'Condition table' is 'FTPUALM1'
  - For the 'Text right aligned' version, the 'Condition table' is 'FTPUALM2'
- Click on 'Close'

### 11.7.4.9. Setting dynamic Attributes for the Control Position Indications

- Single-Click on the CONTROL POSITION INDICATION to be modified
- Handles will appear around the CONTROL POSITION INDICATION
- The Editor shows 'Figure selected : TFAUTO' in its lower left corner
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Dynamic'
- The following dialogue appears:



- Edit the 'Dynamic function' field to be as follows:

Job No.	'Job'	Parameter 1	Parameter 2	Parameter 3	Comment
1	Obj_GetFlag	xxxxxxxxS	Value	65280	Status Word

Where :

- o **xxxxxxxx** = Point ID of the Process Object, to which this Control Position Indication belongs.

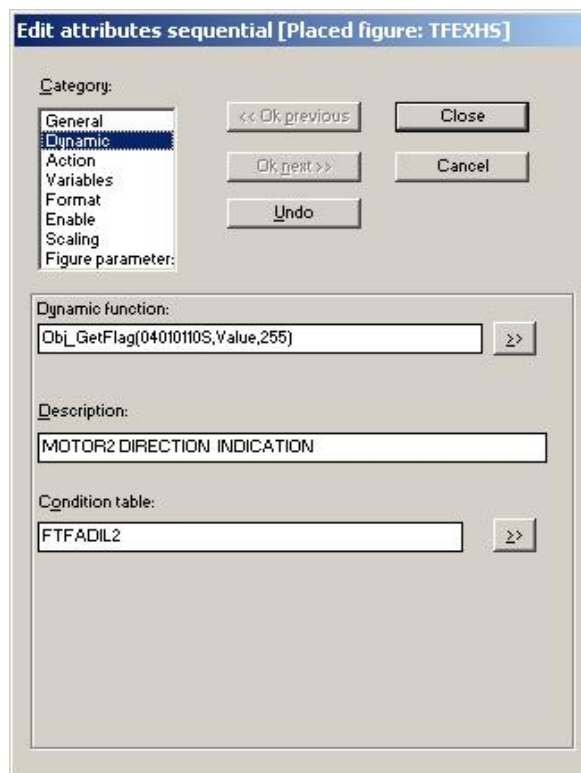
Remark: Parameter 3 contains a Bit Mask (65280), which addresses only the three bits ('Local', 'Manual', 'Auto') from the status word.

- Click on 'Close'

### 11.7.4.10. Setting dynamic Attributes for 'Fan Direction' Indications

These indications are used in conjunction with 2-direction fans (MOTOR2 or MOTOR3). The resulting figure displays the direction of air transport ('Exhaust' or 'Supply') as text indications. These indications are invisible, if the fan is OFF.

- Single-Click on the FAN DIRECTION INDICATION to be modified
- Handles will appear around the FAN DIRECTION INDICATION
- The Editor shows 'Figure selected : 'TFEXHS' in its lower left corner
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Dynamic'
- The following dialogue appears:



- Edit the 'Dynamic function' field to be as follows:

Job No.	'Job'	Parameter 1	Parameter 2	Parameter 3	Comment
1	Obj_GetFlag	xxxxxxxxS	Value	255	Status Word (low Byte)

Where :

- o **xxxxxxxx** = Point ID of the Process Object, to which this Direction Indication belongs.

- Edit the 'Condition Table' field to be as follows:
  - For the 'Text left aligned', MOTOR2 Type, the 'Condition table' is 'FTFADIL2'
  - For the 'Text right aligned', MOTOR2 Type, the 'Condition table' is 'FTFADIR2'
  - For the 'Text left aligned', MOTOR3 Type, the 'Condition table' is 'FTFADIL3'
  - For the 'Text right aligned', MOTOR3 Type, the 'Condition table' is 'FTFADIR3'
- Click on 'Close'

## 11.7.5. Setting dynamic Attributes for Control Windows

A Control Window in IMAC Lean consists of the following components:

- A Figure Condition Table, that controls the switching between the different appearances of the control window, depending on the object currently being controlled.
- A set of n figures, representing the n different appearances of the control window (one figure for each possible type of control).

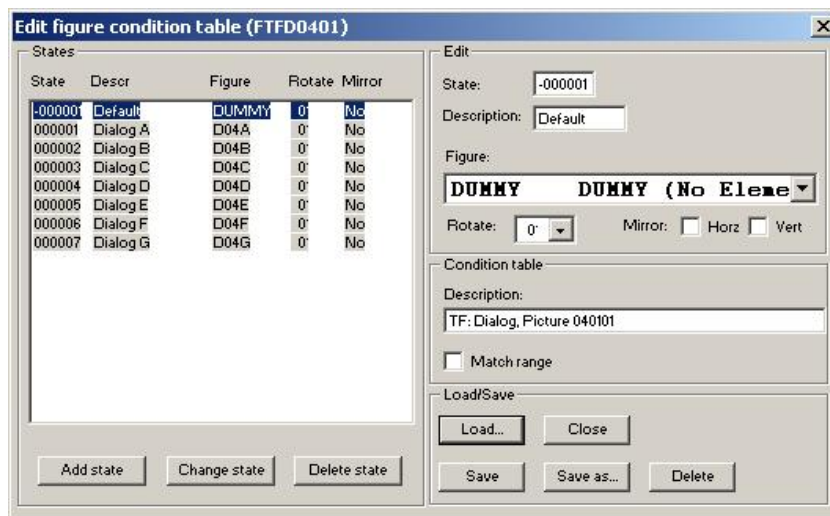
As a fixed rule, these objects have to be copied / renamed for each GD, that has control functions in it. In other words, these objects have to be individual for each GD. Otherwise there will be functional disturbances during change of GDs or if 'Split Screen' is being used.

### 11.7.5.1. Copying of required Control Window Components

The following instructions show, how the existing Control Window template from GD '040101' is being copied for use in a new GD:

#### A. Copy the Figure Condition Table

- Open the template DG '040101' or any other GD in the AbvEditor.
- Click on 'Setup' → 'Figure Condition Tables'
- Click on 'Load'
- Select 'FTFD0401' in the list shown (this is the figure condition table used by the template GD '040101').
- Click on 'OK'
- The following dialogue appears:



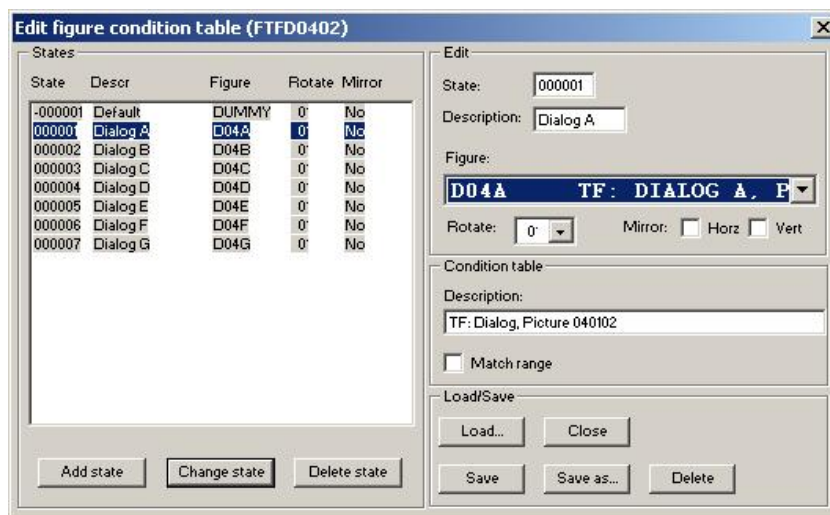
- Edit the 'Description Text' field to contain the GD ID (Picture name) for which the new control window is meant.
- Click on 'Save As'
- In this new dialogue Enter a new name, that does not yet exist (e.g. 'FTFD0402'). The 'Prefix 'FTFD' should be preserved to keep all condition tables of this type together. The 'Postfix xxxx' can be freely selected, but should if possible be identical to the GD ID for which the new control window is meant. Please observe, that the length of the condition table name is restricted to 8 Characters. If your GD names are longer than 4 Characters, the name of the figure condition table will have to receive a shortened form of the name of the GD ID instead.
- Click on 'OK'
- Click on 'Close'

### B. Copy the Figures used in the Figure Condition Table

- In the AbvEditor Click on 'File' → 'Open'
- Select 'Figure from the Type Como Box.'
- Select the name of the first figure in the Figure Condition Table (see section A above). In the IMAC L Template this is 'D04A'.
- Click on 'OK'
- Click on 'Edit' → 'Information...'
- Edit the 'Description' field to contain the GD ID (Picture name) for which this figure is used.
- Click on 'OK'
- Click on 'File' → 'Save As'.
- In this new dialogue Enter a new figure name, that does not yet exist (e.g. 'D0402A'). Please observe, that the length of the figure name is restricted to 6 Characters. The Prefix 'D' is reserved for figures in control windows. The 'Postfix' letter (here 'A') is an ascending index within the control window figure group of one window. The medium part of the name (here '0402') has to be unique for each GD having a control window.
- Click on 'OK'
- Repeat these steps in B. for all figures used in the figure condition table (see section A.)

### C. Modify the new Figure Condition Table to refer to the new figures

- Open the template DG '040101' or any other GD in the AbvEditor.
- Click on 'Setup' → 'Figure Condition Tables'
- Click on 'Load'
- Select the new figure condition table name as entered in section A. (in the example the name is 'FTFD0402').
- Click on 'OK'
- The following dialogue appears:



- In the left 'States' window click on the entry with 'State' = 000001' ('Dialog A').
- From the 'Figure' Combo Box in the upper right window select the newly created figure (from section B. above) - with the index letter 'A'. In our example this action will replace the figure 'D04A' from the template with 'D0402A'
- Click on 'Change State'
- Replace all other figures in this condition table accordingly (repeat last 3 steps 'n' times').
- Click on 'Save'
- Click on 'Close'



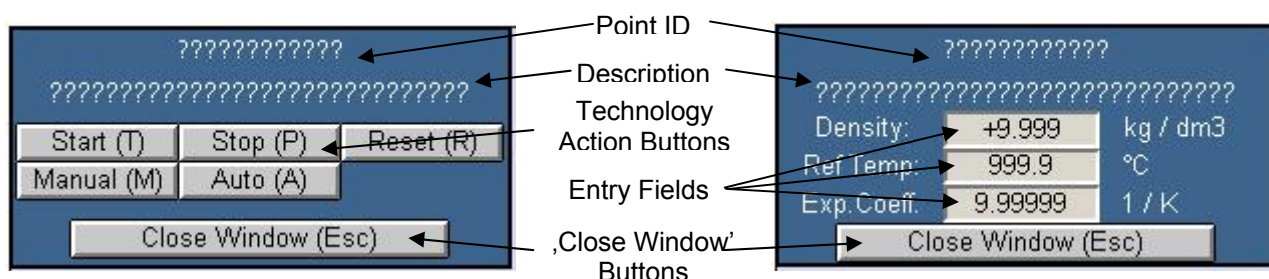
**11.7.5.2. Setting dynamic Attributes of the new Control Window Figures****Important Remark:**

Before continuing with this step, the individual 'SYSTEM OBJECTS' (see → 11.7.2. System Objects required for each GD on page 209) and 'REFERENCE OBJECTS' (see → 11.7.3. Reference Objects required for each GD on page 210) required for the newly created GD have to be created and must be Available in AvEdit.

The references to the names of these 'System' and 'Reference' objects made in this section are identical to the ones used in the sections describing the setup of these data objects ('zzzzzz'). This description assumes, that the suggested naming scheme for these objects is being consequently used.

Basically during the modification of these figures for the control windows, only the references to these objects (belonging to the objects of the old GD) have to be replaced by references to the data objects belonging to the new GD. All other settings (jobs used etc.) will have to stay as they are.

The control window figures consist of the following basic components:



The following sequence ('a' to 'f') has to be executed for each of the 'n' newly created figures in the new figure condition table:

**a.) Text Field 'Point ID'**

- In the AbvEditor Click on 'File' → 'Open'
- Select 'Figure from the Type Como Box.
- Select the name of the figure to be modified.
- Click on the text field in the top of the figure.
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Variables'
- Edit the 'Variables function' to be as follows:

Job No.	'Job'	Parameter 1	Parameter 2	Comment
1	Obj_QueryAttr	XRzzzzzz	PointID	Indication of 'Point ID' in Header

Where :

- o zzzzzz = Name of the new GD.
- Click on 'Close'

**b.) Text Field 'Description'**

- Click on the second text field from the top of the figure.
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Variables'

- Edit the 'Variable function' to be as follows:

Job No.	'Job'	Parameter 1	Parameter 2	Comment
1	Obj_QueryAttr	XRzzzzzz	Description	Indication of 'Description' in Header

Where :

- o zzzzzz = Name of the new GD.
- Click on 'Close'

### c.) 'Technology Action' Buttons

- Click on the first 'Technology Action' button in the figure.
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Action'
- Edit the 'Action function' to be as follows:

Job No.	'Job'	Parameter 1	Parameter 2	Comment
1	Obj_SetVal	XRzzzzzz	(do not modify)	Individual Action Function of Button (Parameter 2 = Bit number to be set)

Where :

- o zzzzzz = Name of the new GD.
- Click on 'Close'
- Repeat this sequence for all 'Technology Action' buttons in the figure.

### d.) 'Entry Fields'

- Click on the first 'Entry Field' in the figure.
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Action'
- Edit the 'Action function' to be as follows:

Job No.	'Job'	Parameter 1	Parameter 2	Comment
1	Obj_SetVal	XRzzzzzzA (first field) XRzzzzzzB (second field) XRzzzzzzC (third field) XRzzzzzzD (fourth field)	(empty)	Input variables A-D.

Where :

- o zzzzzz = Name of the new GD.
- Click on 'Close'
- Repeat this sequence for all entry fields in the figure.

### Remark:

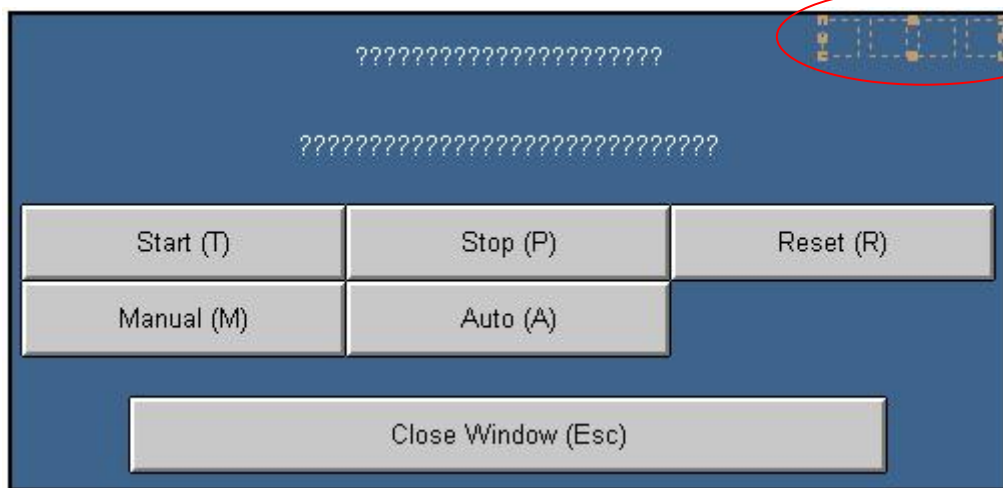
The connection, which 'real' variable is behind each of the input fields is being done in the 'ROUTING BUTTON' action, which causes this figure to be displayed. In this button's action functions real MePo Point IDs (of type PV for input !) are assigned to the 'REFERENCE OBJECTS' named XRzzzzzzA to XRzzzzzzD.

### e.) 'Hidden Entry Fields'

If one of the control window figures uses less than 4 (visible) entry fields, it has to contain the remaining (unused) entry fields in a hidden style (small, in the background). This means, that each control window contains precisely four entry fields (index letters 'A' to 'D'). In addition to the visible entry fields, the hidden fields have to be modified as well.

The 'hidden' entry fields are allocated in the upper right corner of each figure. Only figures having four real (visible) entry fields will not have / require 'invisible' entry field(s).

Select these fields using a 'rubberband action' with your mouse in the upper right corner. If doing this successfully, the result will look like this (here four selected fields):



- Click on 'Arrange' → 'Move to Foreground'. The fields are now visible.
- Click on one of the now visible fields.
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Action'
- Edit the 'Action function' of the entry fields to be as follows.

Job No.	'Job'	Parameter 1	Parameter 2	Comment
1	Obj_SetVal	XR <del>zzzzzz</del> A (first field) XR <del>zzzzzz</del> B (second field) XR <del>zzzzzz</del> C (third field) XR <del>zzzzzz</del> D (fourth field)	(empty)	Dummy entry fields for those of the four entry fields not being used in this figure

Where :

○ ~~zzzzzz~~ = Name of the new GD.

- Click on 'Close'
- Repeat these steps for all remaining 'now visible' entry fields.
- Select all 'now visible entry fields' (rubberband again).
- Click on 'Arrange' → 'Move to Background'. The fields are now invisible again.

f.) 'Close Window' Button

- Click on the 'Close Window' button in the bottom of the figure.
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Action'
- Edit the 'Action function' to be as follows:

Job No.	'Job'	Parameter 1	Parameter 2	Comment
1	Obj_SetSts	XS <del>zzzzzz</del>	0	Reset Figure Selection
2	Obj_SetSts	XS <del>zzzzzz</del> A	0	Reset blinking ALARM TRIANGLE

Where :

- o ~~zzzzzz~~ = Name of the new GD.
- Click on 'Close'
- Click on 'File' → 'Save' to save the modified figure.

**11.7.5.3. Placing the new Control Window in the new GD**

It is recommended to create the new GD by 'Save as' from an existing GD (e.g. template '040101'). The following description shows how to replace the existing (old) control window with the newly created one:

- Open the new GD in the AbvEditor (in our example '040102').
- Click on the existing control window in the upper right corner (handles will appear).
- Click on 'Edit' → 'Delete'.
- Click on 'Insert' → 'Figure'.
- From the list shown now, select the first of the newly created figures (which has been newly created in section 11.7.5.1. Copying of required Control Window Components, part B.) – this is the figure with the index letter 'A' appended to its name.
- Click on 'OK'
- Drag a rectangle in the GD's client area (anywhere) with the mouse: Click & hold left mouse button → Drag a rectangle → Release left Mouse button.
- Select the newly created control window by clicking into it (handles will appear).
- Click on 'Arrange' → 'Original Size'
- Move the control window to coordinates 'x = 1002, y=0' (position indication in the status bar at the bottom of the editor). TIP: Detail movement is easy: Click & hold on the object and then use the cursor keys on your keyboard to move objects 'pixel by pixel'. Release the mouse button when ready.
- Click on 'File' → 'Save'

#### 11.7.5.4. Setting dynamic Attributes of the new Control Window

- Click on the new Control Window figure placed in the previous section.
- Click on 'Edit' → 'Attributes Sequential'
- Click on the category 'Dynamic'
- Edit the 'Dynamic function' field to be as follows:

Job No.	'Job'	Parameter 1	Comment
1	Obj_Sts	XS <del>zzzzzz</del>	The current figure representation depends on the Control Button Action that sets 'XS <del>zzzzzz</del> ' to a value, that selects the right figure for the object to be controlled.

Where :

○ ~~zzzzzz~~ = Name of the new GD.

- Edit the 'Description' field to be 'Dialog select'
- Edit the 'Condition Table' field to be the figure condition table 'FTFDzzzz' that has been newly created in section 11.7.5.1. Copying of required Control Window Components
- Click on 'Close'
- Click on 'File' → 'Save'

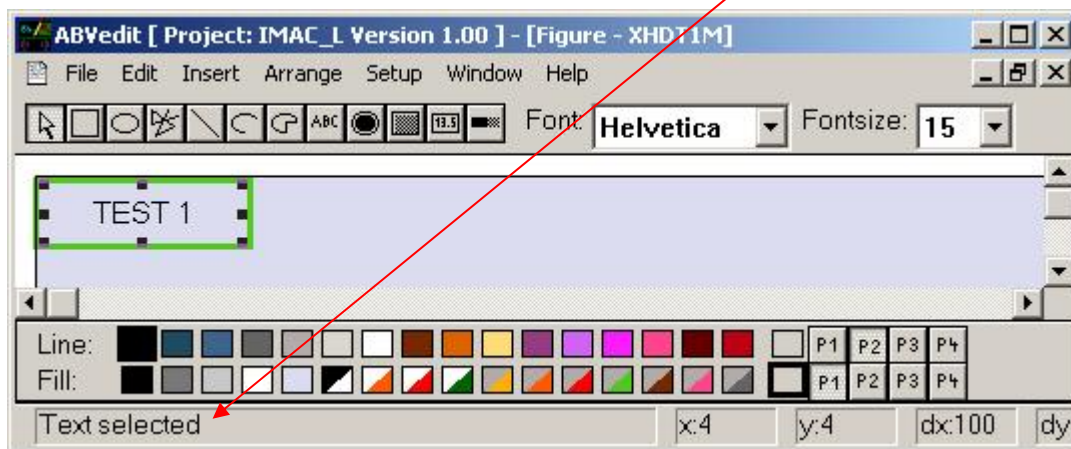
### 11.7.6. Modifying the Header Area

The HEADER AREA of IMAC L contains the buttons for the 16 Technology Areas of IMAC L. These buttons have to be customized for each project. There is only one HEADER AREA in IMAC L.

#### 11.7.6.1. Modifying the 'Text Labels' on the Buttons in the Header Area

To Modify a 'Text Label' on one of the buttons:

- In the AbvEditor Click on 'File' → 'Open'
- Select 'Figure' from the Type Como Box.
- Select the Figure 'XHDTnM' (where n= 1...9, A...G).
- Click on 'OK'
- Single-Click on the center of the figure until 'Text selected' is displayed in the status bar.



- Click on 'Edit' → 'Object'
- Modify the Text as required
- Press 'ENTER' to accept your modifications
- Click on 'File' → 'Save'.
- Click on 'File' → 'Exit'.

Repeat these steps for all other buttons which require modification.

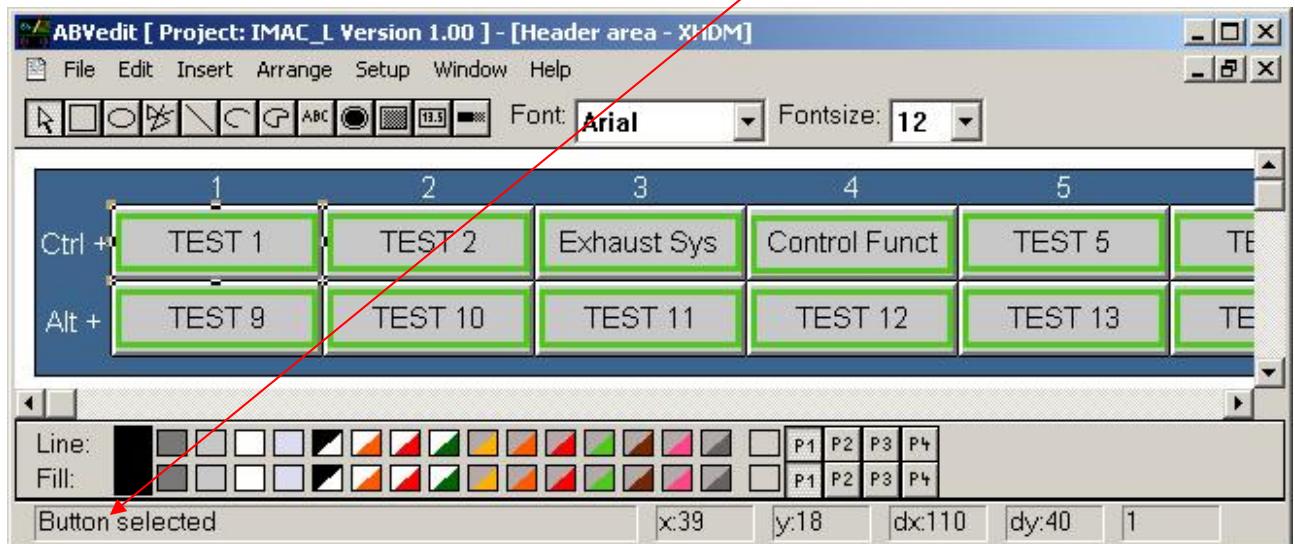
Please observe, that the last Technology Area ('16' – figure 'XHDTGM') is reserved for IMAC. Do not modify !



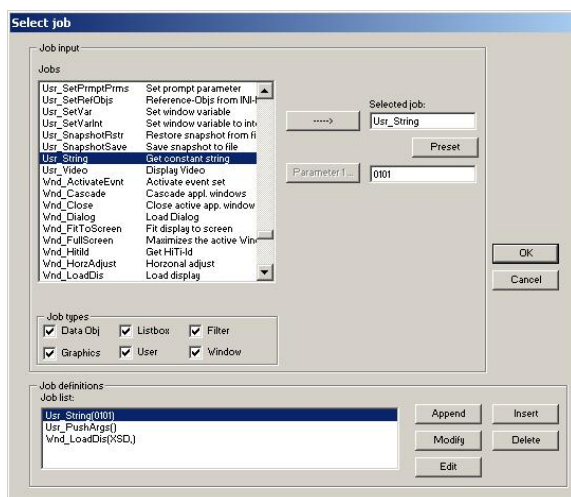
### 11.7.6.2. Modifying the 'Button Actions' of the Buttons in the Header Area

To Modify a 'Button Action' of one of the buttons in the header Area:

- In the AbvEditor Click on 'File' → 'Open'
- Select 'Header Area' from the Type Como Box.
- Select the Header 'XHDM' (Header Merchant).
- Click on 'OK'
- Single-Click on the center of the button until 'Button selected' is displayed in the status bar.



- Click on 'Edit' → 'Attributes Sequential'
- Click on the Category 'Action'
- Click on the '>>' button right of the 'Action Function' field.
- The following dialogue appears:



- The lower part of this dialogue contains a job list. This job list has to be modified according to the tables below in this section.
- To modify an existing job, double click on it. Then modify the job parameters in the upper right corner of the dialogue to fit the requirements of the list below and click on 'Modify' to finish with this job.
- Existing jobs can be removed with the 'Delete Button', if required.
- New jobs can be added with the 'Append' or 'Insert' buttons if required.
- Modify all other jobs as necessary (see tables below in this section).
- Click on 'OK' if finished with all jobs.



- Click on 'Close' to finish editing of the 'Action' of the button.
- Edit other buttons in the HEADER AREA, if required
- Click on 'File' → 'Save'.
- Click on 'File' → 'Exit'.

For a button, which directly opens one of your GRAPHICS DISPLAYS (GDs), the job list has to be as follows:

Job No.	'Job'	Parameter 1	Comment
1	Wnd_LoadDis	aaaaaa	Command to load a 'GD' named 'aaaaaa'

Where :

- o aaaaaa = Name of the GD

For a button, which opens a SECTION DATA list, the job list has to be as follows:

Job No.	'Job'	Parameter 1	Comment
1	Usr_String	bbbb	Load the Section Data list name 'bbbb'.
2	Usr_PushArgs	-	Push name to the Stack
3	Usr_LoadDis	XSD	Load the display 'XSD' (section data) with the section name 'bbbb' taken from the stack.

Where :

- o bbbb = Name of the Section, for which the list is to be opened.

### 11.7.7. Copying / Modifying the Menu Areas

Every Graphics Display (GD) in IMAC L requires its own 'Menu Area'. So if a GD is being copied to a new GD name, a new Header Area has to be copied as well. Otherwise compilation warnings will occur ('Header not found...'). The name of a GD and its Menu Area have to be identical.

The Menu Area is used to make up the 'Tree Structure' of the HMI in IMAC L:

- The root of this tree is the 16 Technology Areas. Each of these buttons branches to one GD (or list). This is the first level of the HMI tree.
- In Each of these 16 GDs the Header Area has to have one button for each GD, that is on the same level of this branch of the tree (within the first level of the HMI tree).
- Optionally there can be one button in each Header Area to navigate to the 'entry point GD' of the next (second) level of the tree.
- Within the next level, this continues as with the first level.

Using this method, HMI trees of any depth of structuring are theoretically possible. Common are one tree level (for simple structures) or two tree levels (for more complex structures).

To copy a Header Area from an existing GD, proceed as follows:

- In the AbvEditor Click on 'File' → 'Open'
- Select 'Menu Area' from the Type Como Box.
- Select the name of the existing Header, which is to be copied.
- Click on 'OK'
- Click on 'File' → 'Save As'.
- Enter a new name in the 'Name' field (has to be identical with the name of the new GD).
- Enter a new description in the 'Descr.' field
- Click on 'Edit' → 'OK'
- Click on 'File' → 'Exit'.

Important Remark:

Please observe, that the arrangement of buttons in the Menu Areas has to be consistent between the Menu Areas within one system regarding the arrangement, size and use of buttons (e.g. 'Back' button always at the same position). Otherwise these buttons will jump 'back & forth', if GDs are changed. Therefore it is recommended to create one Menu Area for one project with the maximum number of 'Menu Buttons' and fixed positions for all of them. All Menus Areas in the System will then just be copies of this Menu, but with some buttons deleted and for the other buttons only the buttons labels and functions are customized for each GD, while their positions are fixed.

A good 'Menu Area' design is one of the keys for a good 'Touch and Feel' of a system, while a sloppy 'Menu Area' design gives the impression, that things are not available (although in fact they are, but the operator does not find them immediately)

To create a new button (copy & paste) in a Menu Area, proceed as follows:

- Click on an existing button in the Menu Area
- Click on 'Edit' → 'Copy'
- Click on 'Edit' → 'Paste'
- Move the new Button to the desired position

To modify the button label of a button in the Menu Area, proceed as follows:

- Click on the button to be modified
- Single-Click on the center of the button until 'Button selected' is displayed in the status bar.
- Click on 'Arrange' → 'Ungroup'
- The Editor shows the indication 'Several primitives selected' in its lower left corner :
- Single-Click on the center of the button until 'Text selected' is displayed in the status bar.
- Click on 'Edit' → 'Object'
- Modify the Text as required
- Press 'ENTER' to accept your modifications
- Use the mouse to catch both the 'button' and the 'text field' using a 'click & drag' motion around these two objects. Be careful not to catch any other objects in this motion.
- The Editor again shows the indication 'Several primitives selected' in its lower left corner
- Click on 'Arrange' → 'Label Button'.

The modification of button actions in Menu Areas and the available functions are identical with the buttons in the 'Header Area' Objects' (see → 11.7.6.2. Modifying the 'Button Actions' of the Buttons in the Header Area on page 237).

## 11.8. Ship Specific Settings

The IMAC L GD 'X2' contains in it's topmost area settings for the 'Ships name' and Voyage Data ('Voyage From' / 'To'):

These settings will be used for headers of printed reports like the IMAC L 'EventLog'. This information can be modified from any OS, if the operator is logged in with sufficient rights.



To distribute and synchronize these entry fields between the connected OSs in IMAC L it is necessary to run the 'OS99' PLC program on OS1 (see → '11.5.1. Start Batch to run SoftPLC Program for 'OS 99' on page 200). This SoftPLC program will hold the data from operator input to these fields and distribute them to all other OSs.

If 'OS99' is not running, these fields will work as well, but every OS will have it's individual settings in these fields (no synchronization).

The initial value of these fields (after restart of IMAC L) can be modified in a way, that the 'Ships Name' field will always hold the name of the respective ship (without operator input being necessary). To modify this default value, please proceed as follows:

- Double-Click on the AvET (AlphaVision Engineering Tool) Icon on your Desktop :
  - Click on 'File → Open'
  - Point to file C:\vision\prjAvET\IMAC\_L\I55L\_Template.prj.xml
  - Click on 'Open'
  - Double-Click on the 'Imported' Folder
  - Double-Click on the 'XIsPV.obj.xml' list
- (The middle window will show a list of all PVs in your project)



- Locate the PV '**\_SHIP\_PV**' and double-click on it  
(The right window will show the properties of '**\_SHIP\_PV**')
- In the right window edit the '**InitValue**' field to fit the ship's name.
- Click on the 'Apply' button () in the right window to save the changes to this list.
- Click on the 'Save' button () of AvET.
- Re-build your project according to this manual.

## 12. Glossary

Abbreviation	Explanation
ACB	Alternator Circuit Breaker
AI	Analogue Input (Module)
AIS	Advanced Identification System
ANTS	Automatic Navigation and Track Steering
AO	Analogue Output (Module)
ARPA	Automatic Radar Plotting Aid
BAPT	<i>Bundesamt für Post und Telekommunikation</i> (Nationale Regulierungsbehörde wurde zum 1.1.1998 ins Leben gerufen und bildet die oberste Bundesbehörde für die Fernmelde- und Telekommunikation)
BSH	<i>Bundesamt für Seeschifffahrt und Hydrographie</i> Federal Maritime and Hydrographic Agency of Germany
c/o	change over (e.g. a standby pump)
CPP	Controllable Pitch Propeller
CRT	Cathode-Ray Tube (Monitor with CRT)
DCS	Damage Control Station
DFO	Diesel Fuel Oil
DG	Diesel Generator
DI	Digital Input (Module)
DO	Digital Output (Module)
DO	Diesel Oil
EAS	Extended Alarm System (Duty Engineer Alarm Panels / Watch Panels)
ECDIS	Electronic Chart Display and Information System
ECR	Engine Control Room
EOT	Engine Order Telegraph (System)
ESB	Emergency Switch Board
ET	Extended Terminals
FAT	Factory Acceptance Test
FFP	Fixed Propeller Plant
FMEA	Fault Mode And Effect Analysis
FO	Fuel Oil
GENOP	Generator protection / synchronizing unit and measuring transducer
GD	Graphic Display, Synonyms: Mimic Display, Mimic, Graphic Picture

Abbreviation	Explanation
GMDSS	Global Maritime Distress & Safety System
HAT	Harbour Acceptance Test
HFO	Heavy Fuel Oil
HMI	Human Machine Interface (Man Machine Interface)
I/O	Input / Output (signals from and to the process)
IAMCS	Integrated Alarm Monitoring and Control System
IAS	Integrated Automation System
ICAS	Integrated Control and Alarm System
ICS	Integrated Control System
IEEE	<a href="#">Institute of Electrical and Electronics Engineers, Inc. (USA)</a>
IMAC	Integrated Monitoring, Alarm and Control System
IMCS	Integrated Monitoring And Control System
ISM	International Safety Management Code
LED	Light Emitting Diode
MCA	Maritime and Coastguard Agency (MCA compliance relates to the structure of yachts, its stability, fire fighting and safety equipment and manning requirements)
MCC	Motor Control Centre
MCR	Machinery Control Room → ECR
MDE	Main Diesel Engine
MDO	Marin Diesel Oil (Marin Fuel)
MSB	Main Switch Board
NC (Contact)	Normally closed contact, in conjunction with relays: Contact is open if relay is energized
NEMA	<a href="#">National Electrical Manufactures Association</a>
NMEA	<a href="#">The National Marine Electronics Association</a> (USA)
NO (Contact)	Normally open contact, in conjunction with relays: Contact is closed if relay is energized
NSB	Non Break Power Supply → USB
PCU	Process Control Unit
POD	Podded Drive
PROFIBUS / DP	Process Field Bus / Decentralised Periphery
PTO	Power Take Off (Gear)
RADAR	RAdio Detection And Ranging
SAT	Sea Acceptance Test
SBG	<i>See-Berufsgenossenschaft</i> / German flag authority



Abbreviation	Explanation
SCR	Selective Catalytic Reduction (Exhaust Gas cleaning system)
SG	Shaft Generator / Switch Gear
SCU	Sub Control Unit
SIMOS	Siemens Marine Operating System
SSP	Siemens Schottel Propulser → POD
TC	Thermo Couples
TFT (Display)	Thin Film Transistor (Technology) (Flat Screen Monitor / Display)
UBL	Unbalanced Load (sharing)
UMS	Unmanned Machinery Space
USB	Uninterruptible Power Supply → NSB
USCG	United States (of America) Coast Guard
VDU	Video Display Unit
VPN	Virtual Private Network