

Configuring Syntax 09/2003 Edition

sinumerik

SINUMERIK 840D/810D  
Configuring Syntax

**SIEMENS**



# SIEMENS

## SINUMERIK 840D/810D

### Configuring Syntax

#### Planning Guide

#### Valid for

<i>Control</i>	<i>Software version</i>	
SINUMERIK 840D		6
SINUMERIK 840DE	(Export version)	6
SINUMERIK 840D	powerline	6
SINUMERIK 840DE	powerline (Export version)	6
SINUMERIK 810D		3
SINUMERIK 810DE	(Export version)	3
SINUMERIK 810D	powerline	6
SINUMERIK 810DE	powerline (Export version)	6

09.03 Edition

Preface

Introduction 1

Configurable Lists 2

Display Elements 3

Action and Reaction Routines 4

Accessing Data on NCK / PLC / MMC 5

MMC Variables 6

Data Types and Parameters 7

Appendix A

Index I

# SINUMERIK® Documentation

## Printing history

Brief details of this edition and previous editions are listed below.

The status of each edition is indicated by the code in the "Remarks" columns.

*Status code in the "Remarks" column:*

- A** .... New documentation.
- B** .... Unrevised reprint with new Order No.
- C** .... Revised edition with new status.  
If factual changes have been made since the last edition, this is indicated by a new edition coding in the header.

<b>Edition</b>	<b>Order No.</b>	<b>Remarks</b>
11.01	Only Online	<b>A</b>
09.03	Only Online	<b>C</b>

This manual is also included in the documentation on CD-ROM (**DOCONCD**)

<b>Edition</b>	<b>Order No.</b>	<b>Remarks</b>
03.04	6FC5 298-6CA00-0BG4	<b>C</b>

## Trademarks

SIMATIC®, SIMATIC HMI®, SIMATIC NET®, SIROTEC®, SINUMERIK® and SIMODRIVE® are registered trademarks of Siemens AG. Other names used in this publication may be trademarks, which, if used by third parties for their own means, may violate the rights of their owners.

More information is available on the Internet at:  
<http://www.siemens.com/sinumerik>

This publication was produced with WinWord V8.0 and Designer V7.0 and the documentation tool AutWinDoc.

The reproduction, transmission, or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration or a utility model or design, are reserved.

Other functions not described in this documentation might be executable in the control. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

We have checked the contents of this manual for agreement with the hardware and software described. Nevertheless, differences might exist and therefore we cannot guarantee that they are completely identical. The information given in this publication is reviewed at regular intervals and any corrections that might be necessary are made in subsequent editions. We welcome all recommendations and suggestions.

# Contents

<b>1 Introduction.....</b>	<b>1-15</b>
1.1 Structure of the Documentation .....	1-16
1.2 Terminology – basic mechanisms.....	1-17
<b>2 Configurable Lists.....</b>	<b>2-23</b>
2.1 Configuring lists.....	2-25
2.1.1 BEGIN_END_: List ID, list type.....	2-25
2.1.2 MENU – menu definition.....	2-27
2.1.3 WINDOW – window definition.....	2-28
2.1.4 WINDOW_HEADER .....	2-29
2.1.5 OBJECT_LIST – description of an object list .....	2-31
2.1.6 SOFTKEY_OBJECT_LIST – definition of a soft key object list .....	2-32
2.1.7 ACTION_LIST - definition of an action list.....	2-33
2.1.8 OPEN_LIST - definition of an open list.....	2-34
2.1.9 CLOSE_LIST - definition of a close list .....	2-35
2.1.10 EVENT_LIST - definition of an event list .....	2-35
2.1.11 REACTION_LIST – definition of a reaction list.....	2-39
2.1.12 SOFTKEY_REACTION_LIST – definition of a soft key reaction list .....	2-40
2.1.13 SYSTEM_INIT_LIST – definition of an initialization list.....	2-42
2.1.14 LIMIT_LIST – definition of an input limit value list .....	2-42
2.2 Break and skip functions in lists.....	2-44
2.2.1 BREAK_UNCOND – unconditional break .....	2-44
2.2.2 SKIP_UNCOND – unconditional skip .....	2-44
2.2.3 BREAK_IF – conditional break .....	2-45
2.2.4 SKIP_IF – conditional skip.....	2-47
2.2.5 LABEL – mark jump destination .....	2-50
2.2.6 GOTO_LABEL – jump to jump destination.....	2-51
2.2.7 OB_DO_ACTION_LIST – execute action list .....	2-51
<b>3 Display Elements.....</b>	<b>3-53</b>
3.1 Static display elements .....	3-56
3.1.1 Point - PIXEL (MMC100/EBF) .....	3-56
3.1.2 Dynamic point - PIXEL_DYN (MMC100/EBF).....	3-56
3.1.3 Line - LINE .....	3-57
3.1.4 Dynamic line - LINE_DYN (MMC100/EBF) .....	3-57
3.1.5 Arrowhead - ARROW (MMC100/EBF) .....	3-58
3.1.6 Dynamic arrowhead - ARROW_DYN (MMC100/EBF).....	3-58
3.1.7 Rectangle - RECTANGLE .....	3-59
3.1.8 Dynamic rectangle - RECTANGLE_DYN (MMC100/EBF).....	3-59
3.1.9 Circle - CIRCLE (MMC100/EBF) .....	3-60
3.1.10 Dynamic circle - CIRCLE_DYN (MMC100/EBF) .....	3-61
3.1.11 Arc, sector - ARC (MMC100/EBF).....	3-62
3.1.12 Dynamic arc, sector - ARC_DYN (MMC100/EBF) .....	3-63
3.1.13 Ellipse - ELLIPSE (MMC100/EBF) .....	3-64

3.1.14	Dynamic ellipse - ELLIPSE_DYN (MMC100/EBF)	3-64
3.1.15	Fill pattern - definition - DEF_PATTERN	3-65
3.1.16	Load color pallet - COL_TAB (MMC100/EBF)	3-66
3.1.17	Texts - TEXT, FIXTEXT	3-66
3.1.18	Dynamic texts - TEXT_DYN	3-70
3.1.19	Polymarker definition - DEF_POLYMARKER	3-72
3.1.20	Polymarker definition - DEF_POLYMARKER (only MMC100/EBF)	3-73
3.1.21	Polymarker - POLYMARKER	3-78
3.1.22	Dynamic polymarker - POLYMARKER_DYN (MMC100/EBF)	3-78
3.1.23	SOFT KEY	3-79
3.1.24	SOFTKEY_PRO	3-81
3.2	Dynamic display elements – dialog fields	3-83
3.2.1	Input/output field - IO_FIELD	3-83
3.2.2	Output field - O_FIELD	3-86
3.2.3	PROGRESS_BAR	3-88
3.2.4	TACHOMETER	3-89
3.2.5	Input field - I_FIELD	3-90
3.2.6	Graphic list field for cursor - CUR_PICT_FIELD	3-93
3.2.7	Single and multiple option boxes - CHECK_FIELD	3-94
3.2.8	Edit field /NC – data overview – EDIT_FIELD	3-97
3.2.9	Edit field_32	3-100
3.2.10	Table - TABLE	3-102
3.2.11	Table column - TAB_COLUMN	3-103
3.2.12	Table data element – line entry - TAB_ITEM	3-106
3.2.13	Graphic list field - PICT_FIELD	3-114
3.2.14	Action field - ACTION_FIELD	3-116
3.2.15	Inverse field - INVERSE_FIELD	3-118
3.2.16	Scrollbar - DEF_SCROLL_BAR	3-118
3.2.17	Macro element (sub-object lists) - MACRO	3-120
3.2.18	Dynamic macro element (sub-object lists) - MACRO_DYN	3-121
3.2.19	Progress bar - PROGRESS_BAR (MMC100/EBF)	3-121
3.2.20	Tachometer element - TACHO (MMC100/EBF)	3-122
3.2.21	Bitmaps - PCX (MMC100/EBF)	3-123
3.3	Colors	3-125
3.3.1	Colors for OP 030 and HPU	3-125
3.3.2	Colors for MMC100/UOP	3-125
3.4	Refresh factor – display and data refresh	3-127
3.5	Data conversion	3-128
3.5.1	Data format for the conversion	3-129
3.5.2	CON_TEXT	3-129
3.5.3	CON_TEXT_OFFSET	3-130
3.5.4	CON_TEXT_BOOL	3-131
3.5.5	CON_ASCII	3-131
3.5.6	CON_STRING	3-132
3.5.7	CON_STRING_LIMIT	3-133
3.5.8	CON_DECIMAL	3-134
3.5.9	CON_HEX	3-135
3.5.10	CON_BINARY	3-136
3.5.11	CON_BCD	3-136
3.5.12	CON_BIT	3-137
3.5.13	CON_OFF	3-137

<b>4 Action and Reaction Routines .....</b>	<b>4-139</b>
4.1 Routines that affect lists and objects .....	4-142
4.1.1 NEW_MENU: Opening a new menu .....	4-142
4.1.2 NEW_MENU_NB: Opening a menu, indirect identifier (OP 030) .....	4-142
4.1.3 OPEN_WINDOW: Opening a window .....	4-143
4.1.4 OPEN_WINDOW_NB: Opening a window, indirect identifier.....	4-144
4.1.5 CLOSE_WINDOW: Closing a window.....	4-145
4.1.6 CLOSE_WINDOW_NB: Closing a window.....	4-146
4.1.7 REFRESH_WINDOW: Refreshing a window object list .....	4-147
4.1.8 NEW_SOFTKEY: Activating a new soft key line .....	4-148
4.1.9 NEW_SOFTKEY_ASSIGN: Activating a new soft key bar.....	4-149
4.1.10 OPEN_BRC_LIST: Opening a basis reaction list.....	4-149
4.1.11 CLOSE_BRC_LIST: Closing a basic reaction list.....	4-150
4.1.12 OPEN_EVENT_LIST: Opening an event list.....	4-150
4.1.13 CLOSE_EVENT_LIST: Closing an event list.....	4-151
4.1.14 OPEN_LIMIT_LIST: Opening an input limit value list.....	4-151
4.1.15 CLOSE_LIMIT_LIST: Closing an input limit value list .....	4-153
4.1.16 DRAW_OBJECT: Processing an object list.....	4-153
4.1.17 PROCESS_ACTION_LIST: Processing an action list.....	4-154
4.1.18 DRAW_SOFTKEY: Output of a soft key (not HPU).....	4-155
4.1.19 ACTIVATE_SK_GRAPHIC: Activating the soft key display .....	4-156
4.1.20 DEACTIVATE_SK_GRAPHIC: De-activating the soft key display .....	4-156
4.1.21 ENABLE_SK_VISUALISATION: Withdrawing the soft key display inhibit (06.04.01).....	4-156
4.1.22 SET_RECALL, RESET_RECALL: Deleting RECALL symbols .....	4-157
4.1.23 SET_MORE, RESET_MORE: Deleting the MORE symbol display (MMC100/EBF) .....	4-157
4.1.24 SET_MORE, RESET_MORE, SET_RECALL, RESET_RECALL.....	4-158
4.1.25 D-CLOSE: Closing an input field with value transfer.....	4-158
4.1.26 D_ABORT: Closing an input field without accepting a value.....	4-159
4.1.27 D_GOTO_DIAFIELD: Positioning the cursor to the dialog field .....	4-159
4.1.28 D_GOTO_DIAFIELD_NB: Positioning the cursor to the dialog field .....	4-160
4.1.29 COPY_CURRENT_DIA_ID .....	4-160
4.1.30 COPY_DIA_ID: Saving the actual cursor position in the notepad .....	4-161
4.1.31 SET_WIN_ATTR, RESET_WIN_ATTR: Changing the attribute in a window .....	4-161
4.1.32 D_SET_DIAFIELD_ATTR: Linking-in attributes .....	4-163
4.1.33 D_RESET_DIAFIELD_ATTR: Resetting dialog field attributes .....	4-165
4.1.34 ACTIVATE_DIA_REFR: Updating dialog fields.....	4-167
4.1.35 D_ACTIVATE_ACTION: Processing the action list.....	4-167
4.1.36 NB_DECREMENT: Decrementing the contents of a notepad entry.....	4-168
4.1.37 NB_INCREMENT: Incrementing the contents of a notepad.....	4-168
4.1.38 SET_TXT_NB: Entering into a text variable .....	4-169
4.1.39 APPEND_TXT_NB_TXT: Attaching text to a text variable.....	4-170
4.1.40 APPEND_TXT_NB_TXT_NB: Attaching text variables.....	4-171
4.1.41 CLEAR_TXT_NB: Deleting a text variable .....	4-171
4.1.42 SCROLL_BAR_REFRESH: Refreshing the scrollbar.....	4-172
4.1.43 SET_EVENT_REACTION: Activating/deactivating an event .....	4-173
4.1.44 BREAK_EVENT: Canceling an event processing .....	4-173
4.1.45 SET_ICON_POS: Setting the user icon bar .....	4-175
4.2 Copying and calculation routines .....	4-175
4.2.1 SET_BIT, RESET_BIT, TOGGLE_BIT: Bit operations.....	4-175

4.2.2	SET_BYTE, SET_WORD, SET_LONG, SET_DOUBLE: Setting a value .....	4-176
4.2.3	CALC_UWORD, CALC_LONG, CALC_DOUBLE: Calculating values ..	4-177
4.2.4	CALC_DATA: Calculating with two variables .....	4-178
4.2.5	COPY_DATA: Copying a variable .....	4-179
4.2.6	COPY_DATA_TO_NB: Copying a variable into the notepad .....	4-179
4.2.7	COPY_BLOCK_NCK_NB: Copying variables blockwise (OP 030).....	4-180
4.2.8	CONVERT_DATA_FORMAT: Converting a data format .....	4-181
4.3	General routines.....	4-182
4.3.1	CHANGE_MODE: NC operating mode change) only OP 030) .....	4-182
4.3.2	CHANNEL_SWITCH: Changing the NC channel .....	4-182
4.3.3	BV_LANGUAGE_CHANGE: Toggling between languages (OP 030)...	4-183
4.3.4	SET_MESSAGE, RESET_MESSAGE: Setting, resetting a message ...	4-183
4.3.5	SET_MSG_POS: Setting the position of the message line (MMC100/EBF) .....	4-184
4.3.6	SET_CUR_TXT_POS: Setting the position of the cursor text.....	4-185
4.3.7	SET_INFO, RESET_INFO: Output, delete help symbol .....	4-185
4.3.8	TOOL_SEARCH: Search for tool in the actual TO area (OP 030) .....	4-186
4.3.9	TOOL_CREATE: Create new tool (OP 030) .....	4-186
4.3.10	TOOL_DELETE: Deleting a tool (OP 030) .....	4-187
4.4	Routines to handle part programs.....	4-188
4.4.1	PP_EDIT_OPEN: Opening an edit field for part programs.....	4-188
4.4.2	PP_EDIT_CLOSE: Closing an edit field for part programs .....	4-189
4.4.3	PP_REFRESH: Refreshing an edit field for part programs .....	4-189
4.5	Routines for diagnostics functionality.....	4-190
4.5.1	DG_INIT_ALARM: Initializing for the alarm overview (OP 030) .....	4-190
4.5.2	DG_INIT_MSG: Initializing for the message overview .....	4-191
4.5.3	OPEN_VERSION: Initializing for the NCK version display.....	4-192
4.5.4	CLOSE_VERSION: Closing the NCK version display.....	4-193
4.5.5	DG_INIT_PASSW: Initializing the password dialog.....	4-193
4.5.6	DG_CLOSE_PASSW: Closing the password dialog .....	4-194
4.5.7	DG_SET_PASSW: Setting the password (OP 030).....	4-194
4.5.8	DG_CHG_PASSW: Changing the password for the actual access stage .....	4-195
<b>5</b>	<b>Accessing Data on NCK / PLC / MMC .....</b>	<b>5-197</b>
5.1	NCK variables – address structure .....	5-198
5.2	NCK variable - configuring .....	5-201
5.3	PLC variable – address structure.....	5-204
5.4	PLC variable - configuring.....	5-206
5.5	MMC-variable (notepad entry – address structure .....	5-209
5.6	MMC variable (notepad entry) - configuring .....	5-210
5.7	Access to variables - example .....	5-210



<b>6 MMC Variables</b> .....	<b>6-211</b>
6.1 MMC variables for OP 030.....	6-212
6.1.1 Variables for alarms and messages .....	6-212
6.1.2 Variables for tables (only OP 030).....	6-214
<b>7 Parameter Data Types</b> .....	<b>7-217</b>
<b>A Appendix</b> .....	<b>A-223</b>
A.1 References .....	A-223
<b>I Index</b> .....	<b>I-235</b>
I.1 Keyword index.....	I-235





# Preface

## Structure of the Documentation

The SINUMERIK documentation is organized in 3 parts:

- General Documentation
- User Documentation
- Manufacturer/service documentation

From 09/2001

- SINUMERIK 840D powerline and
- SINUMERIK 840DE powerline

will be available with improved performance. For a list of available **powerline** modules, please refer to the Hardware Description /PHD/ in Section 1.1

SINUMERIK 810D  
powerline

From 12/2001

- SINUMERIK 810D powerline and
- SINUMERIK 810DE powerline

will be available with improved performance. For a list of available **powerline** modules, please refer to the Hardware Description /PHC/ in Section 1.1

This Manual is intended for use by:

- Programmers
- Maintenance and service personnel

### Finding Your Way Around

To help you navigate through the document, in addition to the list of contents, diagrams and tables, the following information is provided in the Appendix:

1. List of Abbreviations
2. References
3. Index

### Qualified personnel

Commissioning and operation of the devices are to be carried out by qualified personnel only. Qualified personnel are defined as persons who are authorized to commission, to ground, and to tag circuits, equipment, and systems in accordance with the established safety practices and standards.

(Additional references)

**References:** //,

### Notes

The following notes used in the documentation are of special significance:

---

#### Note

Always appears in this document where further, explanatory information is provided.

---



---

#### Important

This symbol is used in this documentation to indicate important information that must be observed.

---



---

#### Ordering data supplement

This symbol refers to an ordering data supplement. The described function can only run if the control contains the designated option.

---

### Safety-related information:

This manual contains information which you should observe in order to ensure your own personal safety, as well to avoid material damage. A warning triangle references this safety-related information and, depending on the degree of the potential hazard, is shown as follows:

## Warnings

The following warning notices with varying degrees of significance are used in the document:



---

### Danger

This symbol indicates that death, severe personal injury or substantial property damage **will** result if proper precautions are not taken.

---



---

### Warning

This symbol indicates that death, severe personal injury or substantial property damage **may** result if proper precautions are not taken.

---



---

### Caution

This warning (with a warning triangle) means that minor physical injury **can** occur if the appropriate precautions are not taken.

---

---

### Caution

This symbol (without a warning triangle) indicates that damage to property **may** result if proper precautions are not taken.

---

---

### Notice

This symbol indicates that an undesirable result or state **may** result if proper precautions are not taken.

---

## Technical information

### Trademarks

IBM® is a registered trademark of the International Business Corporation. MS-DOS® and WINDOWS™ are registered trademarks of Microsoft Corporation.



## Notes

# 1

## 1 Introduction

1.1 Structure of the Documentation .....	1-16
1.2 Terminology – basic mechanisms.....	1-17

## 1.1 Structure of the Documentation

The description of the functions for configuring the user interface for the SINUMERIK OP 030 is subdivided into the following documents:

### OP 030

- **SINUMERIK OP 030 Configuring Syntax**  
General description of the configuring syntax
- **SINUMERIK OP 030 Development Kit**  
Description of the environment for the development of a customized user interface
- **SINUMERIK OP 030 Operator's Guide**  
Operator's Guide for the standard operating functions
- **SINUMERIK OP 030 Installation package**  
Guideline for updating the software on a SINUMERIK OP 030

### MMC 100/UOP

SINUMERIK 840D/810D – HMI Embedded Configuring Package  
Installation, configuring and operating

SINUMERIK 840D/810D/FM NC Operator's Guide  
Operator's Guide for the standard operating functions

<b>Audience</b>	This documentation is intended for the machine tool manufacturers who wants to design their own user interface for the OP 030/MMC 100/UOP.
<b>Target</b>	With the aid of the OP 030/MMC 100/UOP Development Kit and the associated description of functions, the machine tool manufacturer is able to: <ul style="list-style-type: none"><li>• Create their own operator interface.</li><li>• Test this user interface on a PC.</li><li>• Transfer the operator interface created to the target hardware and run it.</li><li>• Create a vendor-specific master disk of the modified system for his own service assignments.</li></ul>
<b>Dependencies</b>	The Development Kit used must match the relevant SINUMERIK 840D/810D version.
<b>Requirements</b>	The hardware and software requirements for using the SINUMERIK OP 030/MMC 100/UOP Development Kit are specified in document <b>References:</b> /FBO/, EU, Development Kit /FB0/, IK, Installation Package



**Notation of the configuration language**

The following grammar rules apply:

"|" separates alternatives (if not explicitly expressed as a logical combination),

"[" and "]" bracket optional components,

"..." identifies the optional repetition of parameters.

Parts printed in *italics* are user parameters. Parts printed in **bold** characters are keywords.

## 1.2 Terminology – basic mechanisms

**User interface**

The purpose of the user interface is to enable the user to monitor and control a machine tool.

The machine tool is controlled by the NC (NCK and PLC). Indirect control of the machine tool is therefore possible by controlling the NCK and the PLC.

**NCK variables PLC variables MMC variables (notepad entry)**

The NCK and PLC do not possess an interface which can be directly accessed by the user, but a defined software interface instead. This interface is used by operator panels such as the OP 030 or MMC 100/UOP.

The OP 030/MMC 100/UOP system uses MMC variables which offer the machine tool manufacturer access to the internal variables of the MMC or to his own self-defined MMC variables. MMC variables are managed in so-called 'notepads' (see Section 5.5).

The objective of the machine tool manufacturer using the OP 030/MMC 100/UOP Development Kit is therefore, using the NCK/PLC interface, to design a user interface which provides the closest contact with the machine tool.

The PLC interface is generally defined by the machine tool manufacturer.

The NCK interface is described in the following documentation:

**References:** /LIS/, Lists

**Object display elements**

The layout of the user interface and the output of data on the screen are controlled with the aid of predefined display elements. The display elements are either static (such as graphics composed of lines and rectangles) or dynamic (e.g. display of variables from the NCK).

Display elements are also referred to on the following pages as display objects, window objects or simply objects.

**Dialog boxes for dynamic window elements**

Dynamic display elements dynamically change their state.

The change can either be initiated by user inputs or by value changes in the NCK/PLC or MMC.

Dialog fields are, for example, output fields, input fields, input/output fields, edit fields, single/multiple selection fields, scroll bars, inverted fields, action fields and tables.

### **Process dialog operation sequence**

The process dialog is controlled mainly by the user through direct interaction with the system.

The process dialog can, however, also be influenced by machine tool states (implicitly via PLC and NCK).

The user's main means of control are soft keys and the direct input of values.

**References:** /BA/, Operator's Guide

### **Soft Key**

With the aid of the soft keys it is possible to organize the user interface in a hierarchical structure.

A soft key is presented with a graphical part and a functional part - the latter are the soft key reactions.

### **Window**

When structuring the user interface, static and dynamic display elements can be combined in a group. Windows is used to combine into such groups.

You can display (open) a window and the display elements grouped inside it or remove it from the screen (close it) by performing an action.

The windows are usually opened or closed through user interaction by pressing a soft key.

### **Menu**

Menus are also provided for controlling user interaction. A menu represents the starting point for a whole chain of windows with soft keys and soft key reactions. This produces implicit combinations of different windows in groups.

Menus - like windows - can be opened or closed as a reaction to soft key selections or changes in NCK/PLC states.

It is possible for two different menus to be active at the same time. These menus are referred to below as 'local menu' and 'global menu'.

The global menu is normally used for displays and reactions which are always relevant for all systems.

Depending on the program branch (e.g. operating area), a new menu is always opened as the local menu.

### **Operating area**

The first soft key level is called the main menu.

---

#### **Note**

You need to press the area switchover key F10 to display the basic screen.

---

It generally reflects the main logical subdivision of the user interface. The individual local menus, which are hidden at this level behind the soft key, are therefore also known as the user area. One menu is normally allocated to one user area.

The main subdivision into user areas can be displayed on the soft key with the User Area Key at any time, regardless of the level of nesting where the user is currently located.

The allocation of the user areas to the soft key on the OP 030 can be found in the file `\proj\std\sy_be_sk.c` (not MMC 100).

**Actions and reactions** Values of NCK/MMC/PLC variables can be read, written, initialized, processed and calculated in action and reaction routines.

Reaction and action routines can be assigned to a single display element, window or menu or globally to the system.

Action routines are executed in transition phases, e.g. when menus or windows are opened or closed.

Action routines are used for initializing, saving and restoring variables and internal statuses.

Reaction routines are executed as a reaction to a user action (such as the press of a soft key or other key) or to changes in variable values (indirectly through events).

**Events** The operating sequence is controlled by two factors: user actions (inputs) and status changes on the machine tool.

User actions are manifested in keyboard events (including soft key events and key actuations).

The state of the machine tool can be supervised by means of monitoring functions (event list with BIT\_EVENT, VALUE\_EVENT, WATCH\_EVENT). When the specified state occurs, a defined internal event is generated.

Specific reactions can be triggered by both keyboard events and internal events.

**Lists configuration lists** The configuring system is based on two management objects: the menu and the window.

Actions, reactions, events, soft keys and display elements (objects) can be assigned to each menu and window.

The number of these individual actions, reactions, events, soft keys and display elements is variable from menu to menu and from window to window.

A configuration list (also referred to as a list) is a collection of elements of the same type (e.g. actions, reactions, display elements).

A list is identified by:

- a list type (action list, reaction list, object list, soft key object list, limit list, ...),

- A unique list identifier assigned by the person configuring the system (list identity).
- One start, one end, and
- A variable number of elements with different functionality (list elements).

Each list in the system can be identified uniquely by the list type and list identifier.



---

**Important**

The list identity must be a constant number or a define or Enum element. Calculating operations are not permitted at this point.

---

**List identifiers**

Each list has its own list identifier.

This list identifier must be unique to the whole system and is assigned by the person configuring the system.

For more information on the list identifier, please refer to:

**References:** /FBO/, EU, Development Kit  
/PJE/, HMI Embedded Configuring Package

**List elements  
List entries**

Each list consists of a variable number of the same or different elements, referred to as list elements.

These list elements are identified by their type and list-element-specific parameters.

Details of which elements (actions, reactions, limit values display elements) can be included in which lists are given with the individual elements.

**Configuration data**

Configuring data is the term used to describe the compilable, linkable and thus interpretable user interface description created with the help of the OP 030 Development Kit, this documentation and the OP 030 configuration macros.

The user-readable form of these files is stored in \*.c files. The file form which can be processed by the OP 030 (MMC 100/UOP) system is stored in file proj.dat (sl.dat).

**Standard operating areas**

The user interface and the configuration data are organized in two groups: the standard user areas and the application user areas.

The configuring lists for these two areas are stored in different directories for the OP030.

Only OP030:

SIEMENS supplies the standard user areas both on the OP 030 and, in source form, in the Development Kit.

With HT6, the source files are also provided.

These standard user areas can be customized by the machine tool manufacturer. It should be remembered when customizing the standard user areas that these areas are supplied again on the next system update.

**References:** /FBO/, BA, Operator's Guide

<b>Application operator area</b>	<p>The machine tool manufacturer has the option of extending or replacing part or all of the standard user areas with his own application user area.</p>
<b>Compiling</b>	<p>Configuring files (*.C files) created by the machine tool manufacturer are converted into binary format (*.OBJ) or library format (*.LIB) with the aid of a compiler.</p> <p><b>References:</b> /FBO/, EU, Development Kit /PJE/, HMI Embedded Configuring Package</p>
<b>Linking</b>	<p>The linking process combines the binary configuration data of the standard user areas and application user areas.</p> <p><b>References:</b> /FBO/, EU, Development Kit /PJE/, HMI Embedded Configuring Package</p>
<b>Testing the configured user interface</b>	<p>The user interface generated with the Development Kit can be tested on a PC in simulation mode.</p> <p>The final test must, however, be performed on the OP 030/MMC 100/UOP hardware.</p>
<b>Texts foreign languages</b>	<p>The OP 030/MMC 100/UOP has a feature which allows the user to switch between two languages (or switching between other texts) on-line.</p> <p>The texts are therefore stored separately from the configuration data.</p> <p>Each text version (language) is stored in a separate directory.</p> <p>For fast access, the texts are stored in binary format with the aid of a text converter.</p> <p>Up to two text versions can be transferred to the target run-time system.</p> <p><b>References:</b> /FBO/, EU, Development Kit /PJE/, HMI Embedded Configuring Package</p>
<b>Standard values for colors, window sizes, font sizes</b>	<p>The default definitions for the available colors, the character set used and the standard window sizes, etc. are described in the following documentation:</p> <p><b>References:</b> /FBO/, EU, Development Kit /PJE/, HMI Embedded Configuring Package</p>
<b>Number ranges</b>	<p>The definition of the list areas, event areas and notepad areas used are described in the following documentation:</p> <p><b>References:</b> /FBO/, EU, Development Kit /PJE/, HMI Embedded Configuring Package</p> <p style="text-align: right;">■</p>

## Notes

# 2

## 2 Configurable Lists

2.1	Configuring lists.....	2-25
2.1.1	BEGIN_END_: List ID, list type.....	2-25
2.1.2	MENU – menu definition.....	2-27
2.1.3	WINDOW – window definition.....	2-28
2.1.4	WINDOW_HEADER.....	2-29
2.1.5	OBJECT_LIST – description of an object list.....	2-31
2.1.6	SOFTKEY_OBJECT_LIST – definition of a soft key object list.....	2-32
2.1.7	ACTION_LIST - definition of an action list.....	2-33
2.1.8	OPEN_LIST - definition of an open list.....	2-34
2.1.9	CLOSE_LIST - definition of a close list.....	2-35
2.1.10	EVENT_LIST - definition of an event list.....	2-35
2.1.11	REACTION_LIST – definition of a reaction list.....	2-39
2.1.12	SOFTKEY_REACTION_LIST – definition of a soft key reaction list.....	2-40
2.1.13	SYSTEM_INIT_LIST – definition of an initialization list.....	2-42
2.1.14	LIMIT_LIST – definition of an input limit value list.....	2-42
2.2	Break and skip functions in lists.....	2-44
2.2.1	BREAK_UNCOND – unconditional break.....	2-44
2.2.2	SKIP_UNCOND – unconditional skip.....	2-44
2.2.3	BREAK_IF – conditional break.....	2-45
2.2.4	SKIP_IF – conditional skip.....	2-47
2.2.5	LABEL – mark jump destination.....	2-50
2.2.6	GOTO_LABEL – jump to jump destination.....	2-51
2.2.7	OB_DÖ_ACTION_LIST – execute action list.....	2-51

The user interface of the SINUMERIK OP 030/MMC 100/UOP is defined by freely configurable lists (see also Introduction).

There are separate list directories for both standard operating areas and application operating areas.

For further details about list directories, please see

**References:** /FBO/, EU, Development Kit.

**OP 030 only:**

The standard operating areas include standard dialogs such as, for example, actual-value display, part program overview, R parameters, etc. which normally exist on every OP 030.

---

**Note**

You should not attempt to modify these dialogs.

---

Screen dialogs which deviate from the basic SINUMERIK OP 030 version (i.e. customized dialogs) are defined in the user operating area.

The initialization actions for the standard and application areas are defined in the system initialization list.

Each operating area originates in a menu.

A menu definition block and one or several window definition blocks exist for each menu. Any number of windows can be assigned to a menu providing there is sufficient memory space.

Every list is identified by its

- type (e.g. object list, event list, window definition block, ...) and a
- number (also referred to as ID below).

The number must be unique for each list type throughout the entire system. As a result, the list identities are subject to certain conventions.



## 2.1 Configuring lists

### 2.1.1 BEGIN\_END\_: List ID, list type

**Description** Lists are configured in the form of C preprocessor macros arranged in list form. Each list is identified by its

- **List type,**
- **Start and an end identifier**
- **List identity** (abbreviated to *id* below).

An optional number of list elements can be inserted between the start and end identifiers.

Reference to lists is made with the **list pointer** .

If a **list pointer** is used to reference lists from other **configuring source data**, i.e. to **external lists**, then an **external list reference** must be entered at the beginning of the configuring source file for this purpose.

#### Syntax

##### General list definition:

BEGIN_ <i>Listentyp</i> ( <i>id</i> )	Beginning id
END_ <i>Listentyp</i> ( <i>id</i> )	End id
<i>Listentyp</i> _PTR ( <i>id</i> )	List pointer
EXTERN_ <i>Listentyp</i> ( <i>id</i> )	External list reference

##### All list types:

Menu definition	BEGIN_MENU ( <i>id</i> ) END_MENU ( <i>id</i> ) EXTERN_MENU ( <i>id</i> )
Window definition	BEGIN_WINDOW ( <i>id</i> ) END_WINDOW ( <i>id</i> ) EXTERN_WINDOW ( <i>id</i> )
Soft key object list	BEGIN_SOFTKEY_OBJECT_LIST ( <i>id</i> ) END_SOFTKEY_OBJECT_LIST ( <i>id</i> ) SOFTKEY_OBJECT_LIST_PTR ( <i>id</i> ) EXTERN_SOFTKEY_OBJECT_LIST ( <i>id</i> )
Window display object list	BEGIN_OBJECT_LIST( <i>id</i> ) END_OBJECT_LIST ( <i>id</i> ) OBJECT_LIST_PTR ( <i>id</i> ) EXTERN_OBJECT_LIST ( <i>id</i> )
Menu/window open list	BEGIN_OPEN_LIST ( <i>id</i> ) END_OPEN_LIST ( <i>id</i> ) OPEN_LIST_PTR ( <i>id</i> ) EXTERN_OPEN_LIST ( <i>id</i> )
Input limit value list	BEGIN_LIMIT_LIST ( <i>id</i> ) END_LIMIT_LIST ( <i>id</i> ) EXTERN_LIMIT_LIST ( <i>id</i> )
Menu/window close list	BEGIN_CLOSE_LIST ( <i>id</i> ) END_CLOSE_LIST ( <i>id</i> )

	CLOSE_LIST_PTR ( <i>id</i> )
	EXTERN_CLOSE_LIST ( <i>id</i> )
Action list	BEGIN_ACTION_LIST ( <i>id</i> ) END_ACTION_LIST ( <i>id</i> ) ACTION_LIST_PTR ( <i>id</i> ) EXTERN_ACTION_LIST ( <i>id</i> )
Event list	BEGIN_EVENT_LIST ( <i>id</i> ) END_EVENT_LIST ( <i>id</i> ) EXTERN_EVENT_LIST ( <i>id</i> )
Reaction list	BEGIN_REACTION_LIST ( <i>id</i> ) END_REACTION_LIST ( <i>id</i> ) REACTION_LIST_PTR ( <i>id</i> ) EXTERN_REACTION_LIST ( <i>id</i> )
Window soft key reaction list	BEGIN_SOFTKEY_REACTION_LIST( <i>id</i> ) END_SOFTKEY_REACTION_LIST ( <i>id</i> ) SOFTKEY_REACTION_LIST_PTR ( <i>id</i> ) EXTERN_SOFTKEY_REACTION_LIST ( <i>id</i> )
System Initialization list	BEGIN_SYSTEM_INIT_LIST ( <i>id</i> ) END_SYSTEM_INIT_LIST ( <i>id</i> ) EXTERN_SYSTEM_INIT_LIST ( <i>id</i> )

**Parameters**      *id*      List identity - unique identifier of list in numerical form. It is better to define the identities with enums or C preprocessor defines to improve the legibility of the lists. The number ranges are defined in the SINUMERIK OP 030 Development Kit.

**Sequence**      Lists which are referenced by a **list pointer** and located in the same **configuring source file** must be defined by **BEGIN\_*list type*(*id*)** and **END\_*list type*(*id*)** before the list pointer is used.

**Example**

```

/* External definitions for list pointers in other configuring source files */
EXTERN_SOFTKEY_REACTION_LIST (SK_RCL_WIN_START)

/* Definitions for list identities */
enum
{WIN_START =31000,
OB_WIN_START,
SOB_WIN_START}

/* Window display object list - start identifier */
BEGIN_OBJECT_LIST (OB_WIN_START)
..
/* End id of the window display object list */
END_OBJECT_LIST (OB_WIN_START)

/* Soft key object list - start identifier */
BEGIN_SOFTKEY_OBJECT_LIST (SOB_WIN_START)
..
/* End id of the soft key object list */
END_SOFTKEY_OBJECT_LIST (SOB_WIN_START)

/*Window definition */
BEGIN_WINDOW (WIN_START)

```

```

...
/* List pointer for window display object list */
OBJECT_LIST_PTR (OB_WIN_START)

/* List pointer for window soft key response list */
SOFTKEY_REACTION_LIST_PTR (SRC_WIN_START)
...
END_WINDOW (WIN_START)

```

## 2.1.2 MENU – menu definition

<b>Description</b>	<p>A menu definition is the basis for screen dialogs. It does not contain any graphics itself and is therefore not visible on the screen (i.e. a pure management object). However, the menu definition specifies the reference positions and the maximum display area of windows opened within the menu.</p> <p>A distinction is made between global and local menus. A <b>global</b> menu is the higher-level management object for a dialog. Its graphic elements generally remain unchanged. The screenforms that change continuously in the course of a dialog are therefore managed in the <b>local</b> menu.</p>																
<b>Syntax</b>	<pre> <b>BEGIN_MENU</b> (<i>menu_id</i>                <i>attr</i>, <i>txt_id_menu_name</i>, <i>init_win_id</i>,                <i>x</i>, <i>y</i>, <i>w</i>, <i>h</i>, <i>bc</i>)                <b>OPEN_LIST_PTR</b>(<i>opl_id</i>)   <b>NULL</b>,                <b>CLOSE_LIST_PTR</b> (<i>cll_id</i>)   <b>NULL</b>  <b>END_MENU</b> (<i>menu_id</i>) </pre>																
<b>Parameters</b>	<table border="0"> <tr> <td style="vertical-align: top;"><i>menu_id</i></td> <td>Unique identifier of the menu definition block. See also <b>References:</b> /FBO/, EU, Development Kit.</td> </tr> <tr> <td style="vertical-align: top;"><i>attr</i></td> <td><b>0</b> or <b>M_CLEAR_BACKGROUND</b> When closing the menu, the menu background (<i>w,h</i> width and height) in the specified background color (<i>bc</i>) is deleted.</td> </tr> <tr> <td style="vertical-align: top;"><i>txt_id_menu_name</i></td> <td>Text number for the menu identifier, always <b>0</b> in current version</td> </tr> <tr> <td style="vertical-align: top;"><i>init_win_id</i></td> <td>ID of window to be opened when the menu is activated 0 if no window is to be opened.</td> </tr> <tr> <td style="vertical-align: top;"><i>x, y</i></td> <td>Menu origin; physical position of menu in pixels in relation to the top, left-hand screen corner (= 0, 0). The position of windows opened afterwards refers to this origin.</td> </tr> <tr> <td style="vertical-align: top;"><i>w, h</i></td> <td>Width and height of the menu area in pixels.</td> </tr> <tr> <td style="vertical-align: top;"><i>bc</i></td> <td>Background color. Colors and gray levels are dependent on the system (refer to Section 3.3 for valid values).</td> </tr> <tr> <td style="vertical-align: top;"><i>opl_id</i></td> <td>ID on a menu opening list that is executed when the menu is activated.</td> </tr> </table>	<i>menu_id</i>	Unique identifier of the menu definition block. See also <b>References:</b> /FBO/, EU, Development Kit.	<i>attr</i>	<b>0</b> or <b>M_CLEAR_BACKGROUND</b> When closing the menu, the menu background ( <i>w,h</i> width and height) in the specified background color ( <i>bc</i> ) is deleted.	<i>txt_id_menu_name</i>	Text number for the menu identifier, always <b>0</b> in current version	<i>init_win_id</i>	ID of window to be opened when the menu is activated 0 if no window is to be opened.	<i>x, y</i>	Menu origin; physical position of menu in pixels in relation to the top, left-hand screen corner (= 0, 0). The position of windows opened afterwards refers to this origin.	<i>w, h</i>	Width and height of the menu area in pixels.	<i>bc</i>	Background color. Colors and gray levels are dependent on the system (refer to Section 3.3 for valid values).	<i>opl_id</i>	ID on a menu opening list that is executed when the menu is activated.
<i>menu_id</i>	Unique identifier of the menu definition block. See also <b>References:</b> /FBO/, EU, Development Kit.																
<i>attr</i>	<b>0</b> or <b>M_CLEAR_BACKGROUND</b> When closing the menu, the menu background ( <i>w,h</i> width and height) in the specified background color ( <i>bc</i> ) is deleted.																
<i>txt_id_menu_name</i>	Text number for the menu identifier, always <b>0</b> in current version																
<i>init_win_id</i>	ID of window to be opened when the menu is activated 0 if no window is to be opened.																
<i>x, y</i>	Menu origin; physical position of menu in pixels in relation to the top, left-hand screen corner (= 0, 0). The position of windows opened afterwards refers to this origin.																
<i>w, h</i>	Width and height of the menu area in pixels.																
<i>bc</i>	Background color. Colors and gray levels are dependent on the system (refer to Section 3.3 for valid values).																
<i>opl_id</i>	ID on a menu opening list that is executed when the menu is activated.																

*cll\_id* ID on a menu close list that is executed when the menu is closed.

**Note** The menu definition block only ever contains one element (one-dimensional list) and always has a predefined length.  
Order of execution of lists when a new menu is opened:  
See action routine NEW\_MENU.

### 2.1.3 WINDOW – window definition

**Description** A window definition determines both the appearance and the operating characteristics of a window dialog.  
The definition block includes a reference to a **window display object list** which contains the graphic elements (window fields and static graphics) of the window and may contain references to **window open lists** and a **window close list**. Where window-related soft keys are required, references to a **soft key object list** and a **window soft key reaction list** can also be specified.

**Syntax**

```
BEGIN_WINDOW (win_id
               attr [| attr]*...,
               x, y, w, h, brdc, bc, ch_grp)
OPEN_LIST_PTR (opl_id) | NULL,
CLOSE_LIST_PTR (cll_id) | NULL,
OBJECT_LIST_PTR(obl_id) | NULL,
REACTION_LIST_PTR (rcl_id) | NULL,
SOFTKEY_OBJECT_LIST_PTR (sk_obl_id) | NULL,
SOFTKEY_REACTION_LIST_PTR (sk_rcl_id) | NULL

END_WINDOW (win_id)
```

**Parameters**

*win\_id* Unique identifier of the window definition block.

*attr* Window attributes.  
Individual window attributes can be combined with one another by OR'ing (*attr1*|*attr2*|*attr3*|...).  
You can specify the following attributes:

W\_REFRESH\_LAST\_SOFTKEY  
When the window is closed, the soft key line that was active before the window was opened is re-activated again.

W\_STORE\_CURSOR\_LOCATION  
If further windows containing dialog fields are opened from this window, the current cursor position is saved. In other words, the dialog cursor is displayed again at its original current position when the additional windows are closed.

**W\_FOCUS\_DISABLE**

The window does not have a focus, i.e. the window is displayed without a border.

**W\_OPEN\_AFTER\_OBJ**

The window open list is not executed from the object list until the graphic has been displayed.

<i>x, y</i>	Relative position of the window in pixels in relation to the top left-hand screen corner of the menu to which the window belongs.
<i>w, h</i>	Width and height of the window in pixels.
<i>brdc</i>	Border color of the window. Colors and gray scales are system-dependent (see Section 3.3 for valid settings).
<i>bc</i>	Background color of the window. When the window closes, this color covers the screen area defined by this position, height and width. Colors and gray scales are system-dependent (see Section 3.3 for valid settings).
<i>ch_grp</i>	Different windows can be joined to form a group by this parameter. The relevant channel can be switched over for all variables of the window by the channel switchover function. <b>ZERO</b> is generally used in this case, i.e. all configured windows are assigned to one channel group.
<i>opl_id</i>	Unique identifier for an action list to be executed when the window is opened.
<i>cll_id</i>	Unique identifier for an action list to be executed when the window is closed.
<i>obl_id</i>	Unique identifier for an object list containing the graphic elements of the window. This makes the window visible on the screen.
<i>rcl_id</i>	Unique identifier for a reaction list which determines the reaction to events as they occur.
<i>sk_obl_id</i>	Unique identifier for a soft key object list to be activated when the window opens.
<i>sk_rcl_id</i>	Unique identifier for a soft key reaction list to be activated when the window opens.

**Note**

The window definition block only ever contains one element (one-dimensional list) and always has a predefined length.

Order in which lists are processed when a window is open and closed: See AC\_OPEN\_WINDOW and AC\_CLOSE\_WINDOW or RC\_OPEN\_WINDOW and RC\_CLOSE\_WINDOW.

**2.1.4 WINDOW\_HEADER****Description**

This macro defines what a header looks like in a window (from 06.03.01 onwards). In order that the settings can be applied to the particular graphic

configuration special definitions are available which are listed in the following. The metrics are saved in the files \*.sym and can be modified.

**Syntax****WINDOW\_HEADER** (*win\_id*, *x*, *y*, *w*, *h*, *OBJECT\_LIST\_PTR(obj\_list)*)**Parameters**

<i>win_id</i>	Unique identifier of the window definition block.
<i>x</i>	X position in pixels referred to 640/480 resolution relative to the window origin.  HMI_DEFAULT_X means that the value set in the metric SYM_HEADER_X is used.
<i>y</i>	Y position in pixels referred to 640/480 resolution relative to the window origin.  HMI_DEFAULT_Y means that the value set in the metric SYM_HEADER_Y is used.
<i>w</i>	Width of the header referred to 640/480 resolution in pixels. HMI_DEFAULT_WIDTH means that the header is just as wide as the window.
<i>h</i>	Height of the header referred to 640/480 resolution in pixels. HMI_DEFAULT_HEIGHT means that the value set in the metric SYM_HEADER_HEIGHT is used.
<i>obj_list</i>	Identity of the object list that contains the configuring for the header line. This must either be defined in the same C module further above or in another C module, whereby an external declaration in the form EXTERN_OBJECT_LIST ( <i>obj_list</i> ) is required further above in the C module.

**Example**

In this particular example, the default values are used. This means that when opening, the header is automatically shown in the size and color set in the system metrics. Only then is the object list OB\_REFP\_HEADER executed that outputs a polymarker and the title text.

```
/* Definition of the object list that contains the header */  
BEGIN_OBJECT_LIST(OB_REFP_HEADER)  
    POLYMARKER(29, 12, 5, POLY_DIFF, SYC_W_HEADER_TEXT)  
    TEXT(105, 30, 5, T_MA_MKS_7, CHAR_SET1,  
        SYC_W_HEADER_TEXT)  
END_OBJECT_LIST(OB_REFP_HEADER)  
  
BEGIN_OBJECT_LIST(OB_MA_W_HEADER)  
    WINDOW_HEADER(100, HMI_DEFAULT_X, HMI_DEFAULT_Y,  
        HMI_DEFAULT_WIDTH, HMI_DEFAULT_HEIGHT,  
        OBJECT_LIST_PTR(OB_REFP_HEADER))  
    MACRO(178, 0, 0, OBJECT_LIST_PTR(OB_MA_REFPAX1))  
    IO_FIELD (...  
    ...
```

```

...
END_OBJECT_LIST(OB_MA_W_REFP)

BEGIN_WINDOW(W_MA_REFPOINT)
    0x0000
    X_W_ISTW, Y_W_ISTW
    WIDTH_W_ISTW, HEIGHT_W;ISTW
    SYNC_BK_CLEAR
    SYNC_W_FILL
    NULL
    OPEN_LIST_PTR(OP_MA_W_REFPOINT)
    CLOSE_LIST_PTR(CL_MA_W_REFPOINT)
    OBJECT_LIST_PTR(OB_MA_W_REFP)
    REACTION_LIST_PTR(RC_MA_W_REFPOINT)
    SOFTKEY_OBJECT_LIST_PTR(SOB_MA_W_REFP)
    SOFTKEY_REACTION_LIST_PTR(SRC_MA_W_REFP)
END_WINDOW(W_MA_REFPOINT)

```

### 2.1.5 OBJECT\_LIST – description of an object list

#### Description

An object list contains the graphic and dialog fields belonging to a window as object elements. Object elements are divided into the following main categories:

- **Static display elements:**  
Static display elements are elements that are output once on the screen and remain there until they are overwritten by other elements (e.g. rectangle, lines, ...).
- **Dynamic display elements:**  
Dialog fields (dynamic display elements) provide the means of communication between man and machine. They allow you to input information into the control (NC/PLC or MMC) and visualize the internal values and states of the control. One example is variable field for input and/or output.
- **Image format commands:**  
These elements are not directly visible on the screen, but can be used in the static display elements configured later. One example is the definition of an infill for static display elements.
- **Control commands:**  
These elements can be used while an image is being formatted to skip object elements within an object list as a function of control system values or states. The dialog screen representation may therefore differ depending on the relevant state of the control system.
- **Macros:**  
Macros are references to object lists. Inserting a macro into an object list as an object element causes the entire object list to which the macro refers to be executed when the relevant dialog is activated. Macros may be nested ten deep.

The position of a graphic element on the screen is calculated by adding

- the position of the menu to which the relevant window belongs,
- the position of the window, the position of any macro over it and
- the position configured for the element.

An object list that is executed only when a window is opened does not require a reference in the user list directory.

If the object list is explicitly displayed on the screen, i.e. not because a window is opened but, for example, as the result of an event via the reaction routine DRAW\_OBJECTS, then a reference to the object list must be entered in the user list directory (file *ap\_l\_dir.h*).

**Syntax**

```
BEGIN_OBJECT_LIST (obl_id)
...
/* List of soft keys */
END_OBJECT_LIST (obl_id)
```

**Parameters**

*obl\_id* Unique identifier of soft key object list.

**Note**

Entry in the user list directory (file *ap\_l\_dir.h*) when used explicitly:

```
EXTERN_OBJECT_LIST_PTR (obl_id)
```

## 2.1.6 SOFTKEY\_OBJECT\_LIST – definition of a soft key object list

**Description**

The soft key object list contains all the graphic elements assigned to a soft key line. It is structured in the same way as an object list. However, a soft key object list **must not contain** any dynamic display elements. This applies equally to macros used within a soft key object list.

All coordinate information in the objects refer to the menu origin specified in the menu definition block.

Separating the soft key line from the other graphics in a dialog enables it to be exchanged independently within the dialog. This must be done, for example, if the number of physically existing soft keys is not sufficient for a dialog.

If the soft key object list is not directly assigned to a window by a pointer or if it has to be exchanged for another list due to an event (e.g. using the NEW\_SOFTKEY routine), then a reference to the list must be entered in the user list directory.

**Syntax**

```
BEGIN_SOFTKEY_OBJECT_LIST (sk_obl_id)
...
/* List of static objects */
END_SOFTKEY_OBJECT_LIST (sk_obl_id)
```

**Parameters**

*sk\_obl\_id* Unique identifier of soft key object list.



## 2.1.7 ACTION\_LIST - definition of an action list

**Description** An action list is a list containing any number of **action elements**. The action list is executed in conjunction with an action field (ACTION\_FIELD) in response to an event. It can also be executed like a subroutine within reaction, open or close lists using the PROCESS\_ACTION\_LIST routine.

The following reference to the action list must be entered in the user list directory (*ap\_l\_dir.h*):

```
EXTERN_ACTION_LIST (acl_id)
```

**Syntax**

```
BEGIN_ACTION_LIST (acl_id)
    AC_funktion (ac_id[, par_1, ..., par_n])    /* List of
                                                    action elements */
...
END_ACTION_LIST (acl_id)
```

**Parameters**

<i>acl_id</i>	Unique identifier of the action list.
<i>funktion</i>	Function designation, type of action element.
<i>ac_id</i>	Action element identifier; it must be unique within a C module.
<i>par_1</i> ... <i>par_n</i>	Parameters which must be passed to the action routine. Number, meaning and possible values of parameters are explained further below in the description of individual action elements.

### 2.1.8 OPEN\_LIST - definition of an open list

**Description** An open list contains any number of action elements which can be executed when a menu is activated or a window opened. The relevant menu or window definition block then contains a reference to the desired open list. If the list belongs to a window, then it is possible to configure whether the relevant open list must be executed first when the window opens or whether the window graphic must be displayed (see W\_OPEN\_AFTER\_OBJ window definition attribute).

**Syntax**

```
BEGIN_OPEN_LIST (opl_id)  
AC_... /* List of action elements */  
...  
END_OPEN_LIST (opl_id)
```

**Parameters** *opl\_id* Unique identifier of the open list.

### 2.1.9 CLOSE\_LIST - definition of a close list

<b>Description</b>	A close list is an action list containing any number of action elements (see above) which can be executed when a menu or window is closed. The relevant menu or window definition block then contains a reference to a close list.
<b>Syntax</b>	<pre> <b>BEGIN_CLOSE_LIST</b> (<i>cll_id</i>) AC_...                /* List of action elements */ ... <b>END_CLOSE_LIST</b> (<i>cll_id</i>) </pre>
<b>Parameters</b>	<i>cll_id</i> Unique identifier of the close list.

### 2.1.10 EVENT\_LIST - definition of an event list

<b>Description</b>	<p>The event list is a configurable list containing any number of event elements (<b>Event elements</b>) which must trigger a reaction. An event list is processed by the event handler after it has been activated in an action list, reaction list or initialization list (-&gt; AC_OPEN_EVENT_LIST / RC_OPEN_EVENT_LIST).</p> <p>The event handler checks whether the data specified in an event element has changed in the connected control (NC/PLC). If it has, it triggers an internal event with event code defined in the event element. If the active reaction lists contain a reaction element, which matches the event code, the basic control activates the associated function.</p> <p>Event elements are divided into three categories.</p> <ol style="list-style-type: none"> <li>1) For <b>BIT_EVENT</b> the event handler only reacts to the change of a bit, defined using a bit mask of a piece of data.</li> <li>2) For <b>VALUE_EVENT</b> it reacts to every change made to a value..</li> <li>3) For a <b>WATCH_EVENT</b> a check is made as to whether the specified has changed and the change corresponds to the type and method, specified in the attribute.</li> </ol>
<b>Syntax</b>	<pre> <b>BEGIN_EVENT_LIST</b> (<i>evl_id</i>) ...                /* List of event elements */ <b>BIT_EVENT</b> (<i>ev_id, cycle, ev_code, bit_attr</i>[<i>bit_attr</i>],            <i>v_adr, v_p1, v_p2, v_p3, v_p4</i>) <b>VALUE_EVENT</b> (<i>ev_id, cycle, ev_code, val_attr</i>[<i>val_attr</i>],            <i>v_adr, v_p1, v_p2, v_p3, v_p4</i>) <b>WATCH_EVENT</b> (<i>ev_id, cycle, ev_code, watch_attr</i>[<i>watch_attr</i>], </pre>

*d\_typ, cmp\_val\_d, v\_adr, v\_p1, v\_p2, v\_p3, v\_p4)*

...

**END\_EVENT\_LIST** (*evl\_id*)

**Parameters**

<i>evl_id</i>	Unique identifier of the event list.
<i>ev_id</i>	Unique identifier of event element; it is freely assigned and must be unique within a C module. The numerical range is defined in the SINUMERIK OP 030 Development Kit.
<i>cycle</i>	Time interval within which the configured NC/PLC data must be checked for change; the factor is specified for a fixed time interval of 100ms (7 => 700ms).
<i>ev_code</i>	<p>Basic event code to be applied if the configured event occurs. The user may specify values between 10000 and 19999.</p> <p>The following applies for <b>BIT_EVENT</b>:</p> <p>When a configured bit from a data byte changes, an event with the code 'ev_code + bit no.' is generated.</p> <p>Example:       Bit 0 → ev_code + 0                   Bit 1 → ev_code + 1                   ... → ...</p> <p>Changes to several configured bits at the same time results in the generation of several internal events.</p> <p>The following applies for <b>VALUE_EVENT</b>:</p> <p>If one of the configured conditions is fulfilled, an internal event with code <i>ev_code</i> is generated. If the attribute <b>ADD_VALUE</b> is set, the current value of the NC/PLC data is also added to <i>ev_code</i>.</p> <p>The following applies for <b>WATCH_EVENT</b>:</p> <p>An event with code 'ev_code' is generated for every fulfilled condition.</p>
<i>bit_attr</i>	<p>Attribute word for <b>BIT_EVENTS</b> with the following format: Bit mask (<i>m</i>) and condition attribute (<i>a</i>).</p> <p>Attribute word (bits):       15.....8 7.....0</p> <p style="text-align: center;"><i>aaaaaaaa mmmmmmmm</i></p> <p>Bit 0 ... 7            Bit mask with the bits to be checked</p> <p>Bit 8 ... 15           Condition attribute</p> <p>The bit mask and several condition attributes (but at least one) can be OR'd.</p> <p>Condition attributes:</p> <p>LOW_HIGH</p> <p style="padding-left: 40px;">An event is generated if one of the bits specified by the bit mask changes state from LOW to HIGH.</p> <p>HIGH_LOW</p>

	An event is generated if one of the bits specified by the bit mask changes state from HIGH to LOW.
	<b>IS_HIGH</b>
	An event is generated if one of the bits specified by the bit mask is set.
	<b>IS_LOW</b>
	An event is generated if one of the bits specified by the bit mask is not set.
<i>val_attr</i>	Attribute word for <b>VALUE_EVENTS</b> The following attributes can be OR'd:
	<b>ZERO_TO_NOTZERO</b>
	An event is generated if the content of the configured NC/PLC data changes from zero to a value which is not zero.
	<b>NOTZERO_TO_ZERO</b>
	An event is generated if the content of the configured NC/PLC data changes from a value which is not zero to zero.
	<b>ANY_CHANGE</b>
	An event is generated every time the value of the configured NC/PLC data changes.
	<b>ADD_VALUE</b>
	As an event is generated, the current value of the NC/PLC data is added to the configured event code ( <i>ev_code</i> ).
<i>watch_attr</i>	Attribute word for <b>WATCH_EVENTS</b> The following attributes ( <b>FALSE_TO_TRUE</b> and <b>TRUE_TO_FALSE</b> ) must not be OR'd. Only one of the two attributes must be OR'd with one of the condition attributes described below.
	<b>FALSE_TO_TRUE</b>
	An event is generated if the content of the specified NC/PLC data has changed, thereby fulfilling the condition defined in the condition attributes described below.
	<b>TRUE_TO_FALSE</b>
	An event is generated if the content of the specified NC/PLC data has changed and, as a result, the condition defined in the condition attributes described below is no longer fulfilled (although it was fulfilled before the change).
	One of the following attributes ( <b>condition attributes</b> ) in each case must be OR'd with <b>FALSE_TO_TRUE</b> or with <b>TRUE_TO_FALSE</b> . It is not permissible to OR more than one attribute.

WATCH\_EQUAL (Type 1)

An event is generated if the content of the configured data has changed and its value is equal to the comparison value (*cmp\_val\_d*).

WATCH\_NOT\_EQUAL (Type 2)

An event is generated if the content of the configured data has changed and its value is not equal to the comparison value (*cmp\_val\_d*).

WATCH\_LESS (Type 3)

An event is generated if the content of the configured data has changed and its value is less than the comparison value (*cmp\_val\_d*).

WATCH\_LESS\_EQUAL (Type 4)

An event is generated if the content of the configured data has changed and its value is less than or equal to the comparison value (*cmp\_val\_d*).

WATCH\_GREATER (Type 5)

An event is generated if the content of the configured data has changed and its value is greater than the comparison value (*cmp\_val\_d*).

WATCH\_GREATER\_EQUAL (Type 6)

An event is generated if the content of the configured data has changed and its value is greater than or equal to the comparison value (*cmp\_val\_d*).

<i>d_typ</i>	Data type of data to be monitored (see also Appendix for structure of data formats):
F_BYTE	= char
F_UBYTE	= unsigned char
F_WORD	= short int
F_UWORD	= unsigned short int
F_LONG	= long int
F_ULONG	= unsigned long int
F_DOUBLE	= double
<i>cmp_val_d</i>	Value with which specified data is compared. This value must be a floating-point number (e.g. 15.0) for a WATCH_EVENT.
<i>v_adr</i>	Data identifier for access to NC/PLC data (see Chapter 5).
<i>v_p1</i> ... <i>v_p3</i>	Additional parameters for data access (see Chapter 5).
<i>v_p4</i>	Channel number

**Note**

An event handler can manage up to 5 event lists.

### 2.1.11 REACTION\_LIST – definition of a reaction list

#### Description

The reaction list is a configurable list which may contain any chosen number of **reaction elements**. In contrast to action elements, the elements in the reaction list (reaction elements) are processed (by the basic control) on a purely event-driven basis. Reaction elements therefore differ from action elements in that they require specification of an additional code. The appropriate routine must be called if this code is generated.

In addition to the codes for the soft keys, that are exclusively processed using the menu control codes in key.h such as KEY\_F1 ... KEY\_F8, or KEY\_RECALL (^) and KEY\_MORE (>)), all event codes are permissible.




---

#### Important

If several successive reactions must be generated in response to the same event, the appropriate reaction elements must be arranged sequentially in the reaction list.

As soon as a reaction element with a different event code is configured, interpretation of the reaction list is aborted.

---

Reaction lists can be activated independently of dialog, or for one specific dialog only. Reaction lists that are used independently of dialog are referred to as **basic reaction lists** and dialog-dependent reaction lists as **dialog-specific reaction lists**.

#### a) Use as a dialog-specific reaction list

Exactly one reference to a dialog-specific reaction list exists within each menu. The menu contains the address of the reaction list referred to by the window last opened for the menu (-> window definition, REACTION\_LIST\_PTR).

If the last window opened had no reference to a reaction list, the existing dialog-specific reaction list remains active. If a window possessing a reference to a reaction list is closed, the dialog-specific reaction list that was valid immediately before the same window was opened is re-activated again.

A window that is superimposed on other windows on the screen when it opens always deactivates the current dialog-specific reaction list. This also applies if the superimposed window does not itself have a reference to a reaction list.

If none of the opened windows of a menu has a reference to a reaction list, then no dialog-specific reaction list is active for the menu concerned.

No reference need be included in the user list directory to reaction lists that are used only for specific dialogs. This is because they are already contained in the relevant window definition blocks.

b) Use as a basic reaction list

A basic reaction list is used independently of dialog and active as soon as it is registered with the basic control. It remains valid until it is deactivated again.

A basis reaction list is activated and de-activated via an action element within an action list or initialization list (-> **AC\_OPEN\_BRC\_LIST**, **AC\_CLOSE\_BRC\_LIST**), or via a reaction element within an active reaction list (-> **RC\_OPEN\_BRC\_LIST**, **RC\_CLOSE\_BRC\_LIST**).

The user list directory must contain a reference to a reaction list if this is to be used as the basic reaction list.

All reaction elements defined in the form of C macros and available for configuring are listed and described in Section "Action and reaction routines" with their associated transfer parameters.

**Syntax**

```
BEGIN_REACTION_LIST (rcl_id)
RC_funktion (rc_id,ev_code[, par_1, ..., par_n])
...                               /* List of reaction elements */
END_REACTION_LIST (rcl_id)
```

**Parameters**

<i>rcl_id</i>	Unique identifier of the reaction list.
<i>Function</i>	Function designation, type of the reaction element.
<i>rc_id</i>	ID of a reaction element; it must be unique within a C module.
<i>ev_code</i> the	Code of the event that should be reacted to by processing reaction element. The soft key codes are excluded.
<i>par_1 ... par_n</i>	Parameters that must be passed-on to the reaction routine. The number, significance and possible values of the parameters are explained further below when describing the individual reaction elements.

**Note**

One dialog-dependent reaction list and up to 5 basic reaction lists can be active simultaneously.

### 2.1.12 SOFTKEY\_REACTION\_LIST – definition of a soft key reaction list

**Description**

The soft key reaction lists are configurable lists containing those reaction elements which must be processed when soft keys are actuated. In this case, only soft key events (codes in key.h such as KEY\_F1...KEY\_F8, KEY\_RECALL) may be configured as event codes for these reaction elements.

The reaction elements used in the soft key reaction list are responsible for branching to other dialogs (opening/closing windows, activating new menus),



for initializing new dialogs where necessary and for terminating an active dialog in a defined manner.

Since each soft key reaction list belongs to a very specific screen dialog, only one of these lists can ever be active at one time. It is also important to note that one soft key reaction list should be active at all times as, without it, soft keys cannot be used to branch to other dialogs and thus to exit the dialog.

The soft key reaction list is always assigned to the **local** menu. If a window opened within the global menu contains a reference to a soft key reaction list to be opened, the reference is ignored.




---

#### Important

If several successive reactions must be generated in response to the same event, the appropriate reaction elements must be arranged sequentially in the soft key reaction list.

---

If a soft key reaction list is not directly assigned to a window by a pointer or if it has to be exchanged for another list in response to an event, then a reference to the list must be entered in the user list directory.

#### Syntax

```
BEGIN_SOFTKEY_REACTION_LIST (sk_rcl_id)
RC_funktion (rc_id, sk_ev_code[, par_1, ..., par_n])
...
/* List of reaction elements */
END_SOFTKEY_REACTION_LIST (sk_rcl_id)
```

#### Parameters

<i>sk_rcl_id</i>	Unique identifier of the reaction list.
<i>Function</i>	Function designation, type of the reaction element.
<i>rc_id</i>	ID of a reaction element; it must be unique within a C module.
<i>sk_ev_code</i>	Code of soft key event which must trigger processing of the reaction element.  (Codes in key.h such as KEY_F1...KEY_F8, KEY_RECALL)
<i>par_1 ... par_n</i>	Parameters which must be passed to the reaction routine. Number, meaning and possible values of parameters are explained further below in the description of individual reaction elements.

### 2.1.13 SYSTEM\_INIT\_LIST – definition of an initialization list

**Description** The initialization list, like the action list, comprises action elements that are processed during system initialization.  
An initialization list is always processed as the system is being initialized.  
There is only one initialization list for the OP 030.

**Syntax**

```
BEGIN_SYSTEM_INIT_LIST (sil_id)  
AC_... /* List of action elements */  
END_SYSTEM_INIT_LIST (sil_id)
```

**Parameters** *sil\_id* Unique identifier of system initialization list.

### 2.1.14 LIMIT\_LIST – definition of an input limit value list

**Description** The input limit value list contains only limit value elements. These elements limit the input of values to particular dialog fields (input/output fields). Any values may be entered provided they are within the permissible input range, otherwise they are rejected.  
Unlike an object or reaction list, the input limit value list is not permanently assigned to one particular window and is not therefore activated automatically when a window opens. It must be registered with the system in an action, reaction or open list by means of the AC\_OPEN\_LIMIT\_LIST or RC\_OPEN\_LIMIT\_LIST function. The user list directory must contain a reference to every limit value list.

**Syntax**

```
BEGIN_LIMIT_LIST (ll_id)  
... /* List of limit value elements */  
LIMIT_ELEMENT (le_id, dia_id, check, lim_l, lim_h)  
DYN_LIMIT_ELEMENT (le_id, dia_id, check,  
 v_adr_l, v_p1_l, v_p2_l, v_p3_l,  
 v_adr_h, v_p1_h, v_p2_h, v_p3_h)  
...  
END_LIMIT_LIST (ll_id)
```

**Parameters** *ll\_id* Unique identifier of the input value list.  
*le\_id* ID of the limit value element. This must be unique within a C module for limit value elements.

<i>dia_id</i>	ID of the input/output field that is valid for the limiting. The object list of the window that is assigned the input limit value list must be an input/output field with this identifier.
<i>check</i>	Type of limit value monitoring LIMIT_CHECK_OFF Do not monitor value for limits LIMIT_CHECK_ON Monitor value for upper and lower limits LIMIT_CHECK_PLUS Check whether input value > 0 LIMIT_CHECK_MIN Monitor lower limit only LIMIT_CHECK_MAX Monitor upper limit only
<i>lim_l</i>	Lower limit value as floating-point number
<i>lim_h</i>	Upper limit value as floating-point number
<i>v_adr_l</i>	Data identifier to access data of NC/PLC/MMC, that contains the lower limit value.
<i>v_p1_l</i> ... <i>v_p3_l</i>	Supplementary parameter to access the data that contains the lower limit value.
<i>v_adr_h</i>	Data identifier to access data of NC/PLC/MMC, that contains the upper limit value.
<i>v_p1_h</i> ... <i>v_p3_h</i>	Supplementary parameter to access the data that contains the upper limit value.

## 2.2 Break and skip functions in lists

It is possible to skip elements in lists, either conditionally or unconditionally, using control commands. By the same means, it is possible to abort processing of a list, either conditionally or unconditionally, at any point.

Depending on the list in which one of the control commands described below is used, the list elements are defined as either

- Objects (preceded by **OB\_**) in object and soft key object lists, or
- Action elements (preceded by **AC\_**) in initialization lists, open lists, close lists and action lists or
- Reaction elements (preceded by **RC\_**) in reaction and soft key reaction lists.

### 2.2.1 BREAK\_UNCOND – unconditional break

**Description** Processing of the list is aborted when this object or list element is reached anywhere in the list.  
Particular display items will not be output or certain functions (actions, reactions) not performed as a result.

**Syntax** **OB\_|AC\_|RC\_ BREAK\_UNCOND** (*id* [, *ev\_code*])

**Parameters**

<i>id</i>	Unique identifier of the list element
<i>ev_code</i>	Event code (only relevant for RC_BREAK_UNCOND).

### 2.2.2 SKIP\_UNCOND – unconditional skip

**Description** The configured number of following list elements is skipped when this object or list element is reached anywhere in the list.  
This will ensure that particular display items will not be output or certain functions (actions, reactions) not performed.

**Syntax** **OB\_|AC\_|RC\_ SKIP\_UNCOND** (*id* [, *ev\_code*], *dist*)

**Parameters**

<i>id</i>	Unique identifier of the initialization list.
<i>ev_code</i>	Event code (only relevant for RC_SKIP_UNCOND)
<i>dist</i>	Skip distance; number of subsequent list elements that should be skipped.




---

**Important**

The jump distance always causes the corresponding number of RC list elements to be skipped, irrespective of their event code.

---

### 2.2.3 BREAK\_IF – conditional break

#### Description

When this object or list element is reached anywhere in the list, processing of the list is aborted as a function of a value or state of a data in the control system.

If the abort criterion is fulfilled, display items will not be output or certain functions (actions, reactions) not performed.

Whether or not list processing must be aborted is determined on the basis of a comparison between a variable and a constant, or between two variables. Whether a variable must be compared with a constant or another variable is - like the data format of the compared values - apparent from the respective call.

#### Syntax

##### Compare variable and constant (C\_):

BYTE values (B\_):

**OB\_C\_B\_BREAK\_IF** (*id,cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**AC\_C\_B\_BREAK\_IF** (*id,cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**RC\_C\_B\_BREAK\_IF** (*id, ev\_code, cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

WORD values (W\_):

**OB\_C\_W\_BREAK\_IF** (*id,cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**AC\_C\_W\_BREAK\_IF** (*id,cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**RC\_C\_W\_BREAK\_IF** (*id, ev\_code, cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

LONG values (L\_):

**OB\_C\_L\_BREAK\_IF** (*id,cmp\_val\_l, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**AC\_C\_L\_BREAK\_IF** (*id,cmp\_val\_l, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**RC\_C\_L\_BREAK\_IF** (*id, ev\_code, cmp\_val\_l, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

DOUBLE values (D\_):

**OB\_C\_D\_BREAK\_IF** (*id,cmp\_val\_d, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**AC\_C\_D\_BREAK\_IF** (*id,cmp\_val\_d, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**RC\_C\_D\_BREAK\_IF** (*id, ev\_code, cmp\_val\_d, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

##### Compare two variables (V\_):

BYTE values (B\_):

<b>OB_V_B_BREAK_IF</b>	<i>(id, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
<b>AC_V_B_BREAK_IF</b>	<i>(id, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
<b>RC_V_B_BREAK_IF</b>	<i>(id, ev_code, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
WORD values (W_):	
<b>OB_V_W_BREAK_IF</b>	<i>(id, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
<b>AC_V_W_BREAK_IF</b>	<i>(id, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
<b>RC_V_W_BREAK_IF</b>	<i>(id, ev_code, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
LONG values (L_):	
<b>OB_V_L_BREAK_IF</b>	<i>(id, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
<b>AC_V_L_BREAK_IF</b>	<i>(id, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
<b>RC_V_L_BREAK_IF</b>	<i>(id, ev_code, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
DOUBLE values (D_):	
<b>OB_V_D_BREAK_IF</b>	<i>(id, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
<b>AC_V_D_BREAK_IF</b>	<i>(id, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>
<b>RC_V_D_BREAK_IF</b>	<i>(id, ev_code, v_adr, v_p1, v_p2, v_p3, cmp_op, v2_adr, v2_p1, v2_p2, v2_p3)</i>

**Parameters**

<i>id</i>	Unique identifier of the list element.
<i>ev_code</i>	Code of the event for which the reaction routine should be activated. (Relevant only for RC_... reaction elements).
<i>cmp_val_w, cmp_val_l, cmp_val_d</i>	Constant value (UWORD [e.g. 123], LONG [e.g. 33123L] or DOUBLE [e.g. 123.4567]) with which a configured data is compared. (For variable/constant comparison only).
<i>v_adr</i>	Data ID for accessing a piece of data in the control.
<i>v_p1 ... v_p3</i>	Additional parameters for data access.
<i>v2_adr</i>	Data ID when accessing a second piece of data in the control. (For comparison of two variables only).
<i>v2_p1 ... v2_p3</i>	Additional parameters for accessing the second data.
<i>cmp_op</i>	Comparison operand; the following values are defined:  BITMASK (not with DOUBLE format)

The value of the (first) configured data is ANDED with the constant or value of the second configured data. Abort if the result is >0.

NOT\_BITMASK (not with DOUBLE format)

The value of the (first) configured data is negated and ANDED with the constant or value of the second configured data. Abort if the result is >0.

EQUAL

Abort if the value of the (first) configured data is equal to the constant or value of the second configured data.

NOT\_EQUAL

Abort if the value of the (first) configured data does not equal the constant or value of the second configured data.

LESS

Abort if the value of the (first) configured data is less than the constant or value of the second configured data.

LESS\_EQUAL

Abort if the value of the (first) configured data is less or equal to than the constant or value of the second configured data.

GREATER

Abort if the value of the (first) configured data is greater than the constant or value of the second configured data.

GREATER\_EQUAL

Abort if the value of the (first) configured data is greater than or equal to than the constant or value of the second configured data.

## 2.2.4 SKIP\_IF – conditional skip

### Description

When this object or list element is reached anywhere in the list, a configured number of the list elements following it are skipped, i.e. not interpreted, as a function of a value or status of a data in the control system.

The jump distance always causes the corresponding number of list elements to be skipped, irrespective of their event code.

If the jump criterion is fulfilled, display items will not be output or certain functions (actions, reactions) not performed.

Whether or not elements are skipped is determined on the basis of a comparison between a variable and a constant, or between two variables.

Whether a variable must be compared with a constant or another variable is - like the data format of the compared values - apparent from the respective call.

**Syntax**

Compare variable and constant (C\_):

BYTE values (B\_):

**OB\_C\_B\_SKIP\_IF** (*id, dist, cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**AC\_C\_B\_SKIP\_IF** (*id, dist, cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**RC\_C\_B\_SKIP\_IF** (*id, ev\_code, dist, cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

WORD values (W\_):

**OB\_C\_W\_SKIP\_IF** (*id, dist, cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**AC\_C\_W\_SKIP\_IF** (*id, dist, cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**RC\_C\_W\_SKIP\_IF** (*id, ev\_code, dist, cmp\_val\_w, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

LONG values (L\_):

**OB\_C\_L\_SKIP\_IF** (*id, dist, cmp\_val\_l, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**AC\_C\_L\_SKIP\_IF** (*id, dist, cmp\_val\_l, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**RC\_C\_L\_SKIP\_IF** (*id, ev\_code, dist, cmp\_val\_l, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

DOUBLE values (D\_):

**OB\_C\_D\_SKIP\_IF** (*id, dist, cmp\_val\_d, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**AC\_C\_D\_SKIP\_IF** (*id, dist, cmp\_val\_d, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

**RC\_C\_D\_SKIP\_IF** (*id, ev\_code, dist, cmp\_val\_d, cmp\_op, v\_adr, v\_p1, v\_p2, v\_p3*)

Compare two variables (V\_):

BYTE values (B\_):

**OB\_V\_B\_SKIP\_IF** (*id, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

**AC\_V\_B\_SKIP\_IF** (*id, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

**RC\_V\_B\_SKIP\_IF** (*id, ev\_code, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

WORD values (W\_):

**OB\_V\_W\_SKIP\_IF** (*id, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

**AC\_V\_W\_SKIP\_IF** (*id, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)



**RC\_V\_W\_SKIP\_IF** (*id, ev\_code, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

LONG values (L\_):

**OB\_V\_L\_SKIP\_IF** (*id, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

**AC\_V\_L\_SKIP\_IF** (*id, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

**RC\_V\_L\_SKIP\_IF** (*id, ev\_code, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

DOUBLE values (D\_):

**OB\_V\_D\_SKIP\_IF** (*id, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

**AC\_V\_D\_SKIP\_IF** (*id, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

**RC\_V\_D\_SKIP\_IF** (*id, ev\_code, dist, v\_adr, v\_p1, v\_p2, v\_p3, cmp\_op, v2\_adr, v2\_p1, v2\_p2, v2\_p3*)

#### Parameters

<i>id</i>	Unique identifier of the list element.
<i>ev_code</i>	Code of the event for which the reaction routine should be activated. (Relevant only for reaction elements).
<i>dist</i>	Skip distance; number of subsequent list elements that should be skipped.




---

#### Important

The jump distance always causes the corresponding number of list elements to be skipped, irrespective of their event code.

---

*cmp\_val\_w, cmp\_val\_l, cmp\_val\_d*

Constant value (UWORD [e.g. 123], LONG [e.g. 33123L] or DOUBLE [e.g. 123.4567]) with which a configured data is compared. (For variable/constant comparison only).

*v\_adr* Data ID to access a piece of data in the control.

*v\_p1 ... v\_p3* Additional parameters for data access.

*v2\_adr* Data ID when accessing a second piece of data in the control. (For comparison of two variables only).

*v2\_p1 ... v2\_p3* Additional parameters for accessing the second data.

*cmp\_op* Comparison operand; the following values are defined:

**BITMASK** (not with DOUBLE)

The value of the (first) configured data is ANDed with the constant or value of the second configured data. Jump if the result is >0.

**NOT\_BITMASK** (not with DOUBLE)

The value of the (first) configured data is ANDed with the constant or value of the second configured data. Jump if the result is equal to 0.

**EQUAL**

Jump if the value of the (first) configured data is equal to the constant or value of the second configured data.

**NOT\_EQUAL**

Jump if the value of the (first) configured data is not equal to the constant or value of the second configured data.

**LESS**

Jump if the value of the (first) configured data is less than the constant or value of the second configured data.

**LESS\_EQUAL**

Jump if the value of the (first) configured data is less or equal to than the constant or value of the second configured data.

**GREATER**

Jump if the value of the (first) configured data is greater than the constant or value of the second configured data.

**GREATER\_EQUAL**

Jump if the value of the (first) configured data is greater than or equal to than the constant or value of the second configured data.

## 2.2.5 LABEL – mark jump destination

<b>Description</b>	Definition of a jump destination.
<b>SYNTAX</b>	<b>OB_ AC_ RC_LABEL</b> ( <i>id</i> [, <i>ev_code</i> ])
<b>Parameters</b>	<i>id</i> Unique identifier of element. This identifier is at the same time the identification of the jump destination. <i>ev_code</i> Event code for RC_LABEL.

## 2.2.6 GOTO\_LABEL – jump to jump destination

**Description** Jumps to a label (OB\_IAC\_IRC\_LABEL) within the list. The jump destination can be positioned either before or after the jump function OB\_IAC\_IRC\_GOTO\_LABEL, but must be **within** the same list as the jump function. In reaction lists, the jump destination must also be located in the same event block (reactions to the same event must be listed consecutively). Processing of the list is aborted if the jump destination cannot be located within it, or within the event block.

**Syntax** OB\_IAC\_IRC\_GOTO\_LABEL (*id* [, *ev\_code*])

**Parameters**

<i>id</i>	Unique identifier of element.
<i>label</i>	Identification of the jump destination. This means that a jump is made to the label OB_IAC_IRC_LABEL
<i>ev_code</i>	Code of the event that should be reacted to by processing the reaction element.

**Example** RC\_GOTO\_LABEL (412, KEY\_ENTER, LB\_TEACH\_BREAK)

---

### Note

Endless loops can be configured with GOTO\_LABEL jumps and the function must therefore be used with great care.

To execute GOTO\_LABEL conditionally, the function can be configured in combination with SKIP and BREAK functions.

---

## 2.2.7 OB\_DO\_ACTION\_LIST – execute action list

**Description** The OB\_DO\_ACTION\_LIST is used in the object lists in order to execute action lists. This means that it is possible to initialize objects before they are used or to call character routines for objects that cannot just use pure configuring elements. The action list is called using a pointer (configuring macro ACTION\_LIST\_PTR( )). It is not necessary to make an entry in the list directory.

**Syntax** OB\_DO\_ACTION\_LIST (*id*, ACTION\_LIST\_PTR(*ac\_list\_id*))

**Parameters**

<i>id</i>	Unique identifier of the element within the module.
<i>ac_list_id</i>	Symbol of the executing action list.

**Example**

```
OB_DO_ACTION_LIST  
(1896;ACTION_LIST_PTR(AC_DI_OB_LLW_SOFTKEYS))
```



## 3

**3 Display Elements**

3.1 Static display elements .....	3-56
3.1.1 Point - PIXEL (MMC100/EBF) .....	3-56
3.1.2 Dynamic point - PIXEL_DYN (MMC100/EBF) .....	3-56
3.1.3 Line - LINE .....	3-57
3.1.4 Dynamic line - LINE_DYN (MMC100/EBF) .....	3-57
3.1.5 Arrowhead - ARROW (MMC100/EBF) .....	3-58
3.1.6 Dynamic arrowhead - ARROW_DYN (MMC100/EBF) .....	3-58
3.1.7 Rectangle - RECTANGLE .....	3-59
3.1.8 Dynamic rectangle - RECTANGLE_DYN (MMC100/EBF) .....	3-59
3.1.9 Circle - CIRCLE (MMC100/EBF) .....	3-60
3.1.10 Dynamic circle - CIRCLE_DYN (MMC100/EBF) .....	3-61
3.1.11 Arc, sector - ARC (MMC100/EBF) .....	3-62
3.1.12 Dynamic arc, sector - ARC_DYN (MMC100/EBF) .....	3-63
3.1.13 Ellipse - ELLIPSE (MMC100/EBF) .....	3-64
3.1.14 Dynamic ellipse - ELLIPSE_DYN (MMC100/EBF) .....	3-64
3.1.15 Fill pattern - definition - DEF_PATTERN .....	3-65
3.1.16 Load color pallet - COL_TAB (MMC100/EBF) .....	3-66
3.1.17 Texts - TEXT, FIXTEXT .....	3-66
3.1.18 Dynamic texts - TEXT_DYN .....	3-70
3.1.19 Polymarker definition - DEF_POLYMARKER .....	3-72
3.1.20 Polymarker definition - DEF_POLYMARKER (only MMC100/EBF) .....	3-73
3.1.21 Polymarker - POLYMARKER .....	3-78
3.1.22 Dynamic polymarker - POLYMARKER_DYN (MMC100/EBF) .....	3-78
3.1.23 SOFT KEY .....	3-79
3.1.24 SOFTKEY_PRO .....	3-81
3.2 Dynamic display elements – dialog fields .....	3-83
3.2.1 Input/output field - IO_FIELD .....	3-83
3.2.2 Output field - O_FIELD .....	3-86
3.2.3 PROGRESS_BAR .....	3-88
3.2.4 TACHOMETER .....	3-89
3.2.5 Input field - I_FIELD .....	3-90
3.2.6 Graphic list field for cursor - CUR_PICT_FIELD .....	3-93
3.2.7 Single and multiple option boxes - CHECK_FIELD .....	3-94
3.2.8 Edit field /NC – data overview – EDIT_FIELD .....	3-97
3.2.9 Edit field_32 .....	3-100
3.2.10 Table - TABLE .....	3-102
3.2.11 Table column - TAB_COLUMN .....	3-103
3.2.12 Table data element – line entry - TAB_ITEM .....	3-106
3.2.13 Graphic list field - PICT_FIELD .....	3-114
3.2.14 Action field - ACTION_FIELD .....	3-116
3.2.15 Inverse field - INVERSE_FIELD .....	3-118
3.2.16 Scrollbar - DEF_SCROLL_BAR .....	3-118

## 3.1 Static display elements

3.2.17	Macro element (sub-object lists) - MACRO .....	3-120
3.2.18	Dynamic macro element (sub-object lists) - MACRO_DYN .....	3-121
3.2.19	Progress bar - PROGRESS_BAR (MMC100/EBF) .....	3-121
3.2.20	Tachometer element - TACHO (MMC100/EBF) .....	3-122
3.2.21	Bitmaps - PCX (MMC100/EBF) .....	3-123
3.3	Colors .....	3-125
3.3.1	Colors for OP 030 and HPU .....	3-125
3.3.2	Colors for MMC100/UOP .....	3-125
3.4	Refresh factor – display and data refresh .....	3-127
3.5	Data conversion .....	3-128
3.5.1	Data format for the conversion .....	3-129
3.5.2	CON_TEXT .....	3-129
3.5.3	CON_TEXT_OFFSET .....	3-130
3.5.4	CON_TEXT_BOOL .....	3-131
3.5.5	CON_ASCII .....	3-131
3.5.6	CON_STRING .....	3-132
3.5.7	CON_STRING_LIMIT .....	3-133
3.5.8	CON_DECIMAL .....	3-134
3.5.9	CON_HEX .....	3-135
3.5.10	CON_BINARY .....	3-136
3.5.11	CON_BCD .....	3-136
3.5.12	CON_BIT .....	3-137
3.5.13	CON_OFF .....	3-137

Display elements are display objects contained in **object (OBJECT\_LIST)** or **soft key object lists (SOFTKEY\_OBJECT\_LIST)** which determine the appearance of screen dialogs.

Display elements are classed as either "static" or "dynamic".

**Static elements** are elements without a variable link such as

- Window
- Text
- Pixel (MMC 100/UOP only)
- Line
- Arrowhead (MMC 100/UPO only)
- Rectangle
- Circle (MMC 100/UOP only)
- Arc (MMC 100/UOP)
- Ellipse (MMC 100/UOP only)
- Infill pattern
- Color range (MMC 100/UOP only)
- Soft key (not HPU)
- Icon
- PCX picture (MMC 100/UOP only)

The **dynamic elements (dialog fields)**, which are linked to, and modifiable via, a variable include

Input fields  
Output fields  
Input/output fields  
Edit fields  
Check fields  
Action fields  
Inverse fields  
Tables  
Table columns  
Table entries  
Icon fields  
Scrollbar  
Macro elements

Display element positions are basically relative, i.e. they refer to the position of the coordinate system of the object to which they belong. This may be a macro, or a window whose position, in turn, refers to the position of the higher-level menu.

The position of a display element in relation to the coordinate system of the physical display area is calculated as follows:

$$X_{\text{phys}} = X_{\text{MEN}} + X_{\text{WIN}} [+ X_{\text{MACRO}_1} + \dots + X_{\text{MACRO}_n}] + X_{\text{ELEMENT}}$$

$$Y_{\text{phys}} = Y_{\text{MEN}} + Y_{\text{WIN}} [+ Y_{\text{MACRO}_1} + \dots + Y_{\text{MACRO}_n}] + Y_{\text{ELEMENT}}$$

Predefined values for the character sets used and standard window sizes are described in

**References:** /DK/, SINUMERIK OP 030 Development Kit  
/PK/, SINUMERIK MMC100/UOP Configuring Package

## 3.1 Static display elements

Static display elements, i.e. elements that are not linked to a variable, are output once on the screen when an object list or soft key object list is processed and then remain unchanged.

A soft key object list may contain only soft key elements.

Soft key elements must always be configured in the soft key object list.

### 3.1.1 Point - PIXEL (MMC100/EBF)

**Description** Display a single dot (picture element) on the screen.

**Syntax** **PIXEL** (*id, x, y, color*)

**Parameters**

<i>id</i>	Unique identifier of element.
<i>x, y</i>	Coordinates of the point in pixels relative to the reference position.
<i>color</i>	Color: Colors and gray scales are system-dependent (refer to the Section – <b>Colors</b> for valid values).

### 3.1.2 Dynamic point - PIXEL\_DYN (MMC100/EBF)

**Description** Display a single dot on the screen; its coordinates and color are read from notepads.

**Syntax** **PIXEL\_DYN** (*id, nb\_x, nb\_y, nb\_color*)

**Parameters**

<i>id</i>	Unique identifier of element.
<i>nb_x, nb_y</i>	Numbers of notepads containing the coordinates of the dot in pixels relative to the reference position.
<i>nb_color</i>	Notepad that contains the color. Colors and gray stages depend on the system (refer to Section <b>Colors</b> for valid values).



### 3.1.3 Line - LINE

<b>Description</b>	Output a line on the screen.	
<b>Syntax</b>	<b>LINE</b> ( <i>id, x1, y1, x2, y2, color, style</i> )	
<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>x1, y1</i>	Starting point of the line in pixels, relative to the reference position.
	<i>x2, y2</i>	End point of the line in pixels, relative to the reference position. It must be observed that <i>x2</i> is greater than <i>x1</i> .
	<i>color</i>	Color: Colors and gray stages depend on the system (refer to the Section – <b>Colors</b> for valid values).
	<i>style</i>	Line pattern as 8-bit value. For the numerical value to be specified (preferably, hexadecimal), each bit represents a pixel. For each bit that is set, a pixel is displayed with the color defined in <i>color</i> . All bits that are not set, mean that the associated pixels remain unchanged. A value of 0xff produces a continuous line.
<b>Note</b>	<i>x1</i> must be less than or equal to <i>x2</i> .	

### 3.1.4 Dynamic line - LINE\_DYN (MMC100/EBF)

<b>Description</b>	Display a line on the screen; its positions and color are read from notepads.	
<b>Syntax</b>	<b>LINE_DYN</b> ( <i>id, nb_x1, nb_y1, nb_x2, nb_y2, nb_color, style</i> )	
<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>nb_x1, nb_y1</i>	Numbers of notepads containing the start point of the line in pixels, relative to the reference position.
	<i>nb_x2, nb_y2</i>	Numbers of notepads containing the end point of the line in pixels, relative to the reference position.
	<i>nb_color</i>	Notepad that contains the color. Colors and gray stages depend on the system (refer to Section <b>Colors</b> for valid values).
	<i>style</i>	Line pattern as 8-bit value. For the numerical value to be specified (preferably, hexadecimal), each bit represents a pixel. For each bit that is set, a pixel is displayed with the color defined in <i>color</i> . All bits that are not set, mean that the associated pixels remain unchanged. A value of 0xff produces a continuous line.

**Note** The contents of notepad *nb\_x1* must be less than or equal to the contents of notepad *nb\_x2*.

### 3.1.5 Arrowhead - ARROW (MMC100/EBF)

**Description** Display an arrowhead on the screen.

**Syntax** **ARROW** (*id, x, y, len, angle, color*)

<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>x, y</i>	End point of the arrowhead in pixels, relative to the reference position.
	<i>len</i>	Length of the arrowhead in pixels.
	<i>angle</i>	Angle in degrees (0...360). 0 corresponds to →, counter-clockwise count direction.
	<i>color</i>	Color: Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).

### 3.1.6 Dynamic arrowhead - ARROW\_DYN (MMC100/EBF)

**Description** Display an arrowhead on the screen; its position, length, angle and color are read from notepads.

**Syntax** **ARROW\_DYN** (*id, nb\_x, nb\_y, nb\_len, nb\_angle, nb\_color*)

<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>nb_x, nb_y</i>	Numbers of notepads containing the end point of the arrowhead in pixels, relative to the reference position.
	<i>nb_len</i>	Number of the notepad containing the length of the arrowhead in pixels.
	<i>nb_angle</i>	Number of the notepad containing the angle of the arrowhead.
	<i>nb_color</i>	Notepad that contains the color. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).

### 3.1.7 Rectangle - RECTANGLE

<b>Description</b>	Display a rectangle on the screen.	
<b>Syntax</b>	<b>RECTANGLE</b> ( <i>id, x, y, w, h, fill, color, style</i> )	
<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>x, y</i>	Position of top left-hand corner of the rectangle in pixels relative to the reference position (window, menu).
	<i>w, h</i>	Width ( <i>w</i> ) and height ( <i>h</i> ) of the rectangle in pixels..
	<i>fill</i>	Number of the fill pattern for the rectangle. Using <b>DEF_PATTERN</b> (refer below), the user can define the required fill pattern. Predefined settings are: NOT_FILLED Only the border of the rectangle is displayed in the color defined in <i>color</i> . Only pixels defined in <i>style</i> are taken into account. FILLED The entire area of the rectangle (including border) is displayed in the color defined in <i>color</i> .
	<i>color</i>	Color: Colors and gray stages depend on the system (refer to the Section – <b>Colors</b> for valid values).
	<i>style</i>	Line style for the rectangle border as an 8-bit value. Each bit in the specified numerical value (preferably hexadecimal) represents a pixel. A pixel of the color defined in <i>color</i> is drawn for each set bit. Bits that are not set cause the associated border pixels to remain unchanged. A value of 0xff produces a continuous line.

### 3.1.8 Dynamic rectangle - RECTANGLE\_DYN (MMC100/EBF)

<b>Description</b>	Display a rectangle on the screen; its position, dimensions and color are read from notepads.
<b>Syntax</b>	<b>RECTANGLE_DYN</b> ( <i>id, nb_x, nb_y, nb_w, nb_h, fill, nb_color, style</i> )

## 3.1 Static display elements

<b>Parameter</b>	<i>id</i>	Unique identifier of element.
	<i>nb_x, nb_y</i>	Notepads containing the position of the top, left-hand corner of the rectangle in pixels relative to the reference position (window, menu).
	<i>nb_w, nb_h</i>	Notepads containing the width ( <i>w</i> ) and height ( <i>h</i> ) of the rectangle in pixels.
	<i>fill</i>	Number of the fill pattern for the rectangle.  Using <b>DEF_PATTERN</b> (refer below), the user can define the required fill pattern.  Predefined settings are:  NOT_FILLED  Only the border of the rectangle is displayed in the color defined in <i>color</i> . Only pixels defined in <i>style</i> are taken into account.  FILLED  The entire area of the rectangle (including the border) is displayed in the color defined in <i>color</i> .
	<i>nb_color</i>	Notepad that contains the color. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
	<i>style</i>	Line style for the rectangle border as an 8-bit value. Each bit in the specified numerical value (preferably hexadecimal) represents a pixel. A pixel of the color defined in <i>color</i> is drawn for each set bit. Bits that are not set cause the associated border pixels to remain unchanged. A value of 0xff produces a continuous line.

## 3.1.9 Circle - CIRCLE (MMC100/EBF)

<b>Description</b>	Display a circle on the screen.								
<b>Syntax</b>	<b>CIRCLE</b> ( <i>id, x, y, r, fill, color, style</i> )								
<b>Parameters</b>	<table border="0"> <tr> <td style="vertical-align: top;"><i>id</i></td> <td>Unique identifier of element.</td> </tr> <tr> <td style="vertical-align: top;"><i>x, y</i></td> <td>Position of the center of the circle point in pixels relative to the reference position (window, menu).</td> </tr> <tr> <td style="vertical-align: top;"><i>r</i></td> <td>Radius in pixels.</td> </tr> <tr> <td style="vertical-align: top;"><i>fill</i></td> <td>Number of the fill pattern for the circle.  Using <b>DEF_PATTERN</b> (refer below), the user can define the required fill pattern.  Predefined settings are:  NOT_FILLED</td> </tr> </table>	<i>id</i>	Unique identifier of element.	<i>x, y</i>	Position of the center of the circle point in pixels relative to the reference position (window, menu).	<i>r</i>	Radius in pixels.	<i>fill</i>	Number of the fill pattern for the circle.  Using <b>DEF_PATTERN</b> (refer below), the user can define the required fill pattern.  Predefined settings are:  NOT_FILLED
<i>id</i>	Unique identifier of element.								
<i>x, y</i>	Position of the center of the circle point in pixels relative to the reference position (window, menu).								
<i>r</i>	Radius in pixels.								
<i>fill</i>	Number of the fill pattern for the circle.  Using <b>DEF_PATTERN</b> (refer below), the user can define the required fill pattern.  Predefined settings are:  NOT_FILLED								

Only the border of the circle is displayed in the color defined in *color*. Only pixels defined in *style* are taken into account.

#### FILLED

The entire area of the circle (including border) is displayed in the color defined in *color*.

<i>color</i>	Color: Colors and gray stages depend on the system (refer to the Section – <b>Colors</b> for valid values).
<i>style</i>	Line style for the circle border as an 8-bit value. Each bit in the specified numerical value (preferably hexadecimal) represents a pixel. A pixel of the color defined in <i>color</i> is drawn for each set bit. Bits that are not set cause the associated border pixels to remain unchanged. A value of 0xff produces a continuous line.

### 3.1.10 Dynamic circle - CIRCLE\_DYN (MMC100/EBF)

**Description** Display a circle on the screen; its position, radius and color are read from notepads.

**Syntax** **CIRCLE\_DYN** (id, nb\_x, nb\_y, nb\_r, fill, nb\_color, style)

<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>nb_x, nb_y</i>	Numbers of notepads containing the center point of the circle in pixels relative to the reference position (window, menu).
	<i>nb_r</i>	Number of the notepad containing the radius in pixels.
	<i>fill</i>	Number of the fill pattern for the circle.  Using <b>DEF_PATTERN</b> (refer below), the user can define the required fill pattern.  Predefined settings are:  NOT_FILLED  Only the border of the dynamic circle is displayed in the color defined in <i>color</i> . Only pixels defined in <i>style</i> are taken into account.  FILLED  The entire area of the dynamic circle (including border) is displayed in the color defined in <i>color</i> .
	<i>nb_color</i>	Number of notepad containing the color. Colors and gray scales are system-dependent (refer to Section <b>Colors</b> for valid values).

*style* Line style for the circle border as an 8-bit value. Each bit in the specified numerical value (preferably hexadecimal) represents a pixel. A pixel of the color defined in *color* is drawn for each set bit. Bits that are not set cause the associated border pixels to remain unchanged. A value of 0xff produces a continuous line.

### 3.1.11 Arc, sector - ARC (MMC100/EBF)

**Description** Display an arc/sector on the screen.

**Syntax** **ARC** (*id, xs, ys, xe, ye, r, direct, sector, color, style*)

**Parameters**

<i>id</i>	Unique identifier of element.
<i>xs, ys</i>	Start position of circular arc/sector in pixels relative to the reference position (window, menu).
<i>xe, ye</i>	End position of circular arc/sector in pixels relative to the reference position (window, menu).
<i>r</i>	Radius in pixels.
<i>direct</i>	Direction in which arc is to be drawn. Valid values are: CLOCKWISE: Coordinates of the start and end point are linked in a clockwise direction. COUNTER_CLOCKWISE: Coordinates of the start and end point are linked in an anti-clockwise direction.
<i>sector</i>	Flag specifying whether a sector must be drawn or just the circular arc. Valid values are: FALSE: Only a circular arc is drawn. TRUE: A sector is drawn, i.e. the start and end coordinates are linked to the calculated center point.
<i>color</i>	Color: Colors and gray stages depend on the system (refer to the Section – <b>Colors</b> for valid values).
<i>style</i>	Line style for the circular arc/sector as an 8-bit value. Each bit in the specified numerical value (preferably hexadecimal) represents a pixel. A pixel of the color defined in <i>color</i> is drawn for each set bit. Bits that are not set cause the associated border pixels to remain unchanged. A value of 0xff produces a continuous line.

### 3.1.12 Dynamic arc, sector - ARC\_DYN (MMC100/EBF)

**Description** Display a circle/sector on the screen; its position, radius and color data are read from notepads.

**Syntax** **ARC\_DYN** (*id, nb\_xs, nb\_ys, nb\_xe, nb\_ys, nb\_r, direct, sector, nb\_color, style*)

**Parameters**

<i>id</i>	Unique identifier of element.
<i>nb_xs, nb_ys</i>	Numbers of the notepads containing the start position of the circular arc/sector in pixels relative to the reference position (window, menu).
<i>nb_xe, nb_ys</i>	Numbers of notepads containing the end position of the circular arc/sector in pixels relative to the reference position (window, menu).
<i>nb_r</i>	Number of the notepad containing the radius in pixels.
<i>direct</i>	Direction in which arc is to be drawn. Valid values are: CLOCKWISE: Coordinates of the start and end point are linked in a clockwise direction. COUNTER_CLOCKWISE: Coordinates of the start and end point are linked in an anti-clockwise direction.
<i>sector</i>	Flag specifying whether a sector must be drawn or just the circular arc. Valid values are: FALSE: Only a circular arc is drawn. TRUE: A sector is drawn, i.e. the start and end coordinates are linked to the calculated center point.
<i>nb_color</i>	Number of notepad containing the color. Colors and gray scales are system-dependent (refer to Section <b>Colors</b> for valid values).
<i>style</i>	Line style for the circular arc/sector as an 8-bit value. Each bit in the specified numerical value (preferably hexadecimal) represents a pixel. A pixel of the color defined in <i>color</i> is drawn for each set bit. Bits that are not set cause the associated border pixels to remain unchanged. A value of 0xff produces a continuous line.

### 3.1.13 Ellipse - ELLIPSE (MMC100/EBF)

<b>Description</b>	Display an ellipse on the screen.	
<b>Syntax</b>	<b>ELLIPSE</b> ( <i>id, x, y, rx, ry, fill, color, style</i> )	
<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>x, y</i>	Ellipse center point in pixels relative to the reference position (window, menu).
	<i>rx, ry</i>	Radius in x, and y directions in pixels.
	<i>fill</i>	Number of the fill pattern for the ellipse. Using <b>DEF_PATTERN</b> , the user can define the required fill pattern. Predefined settings are: NOT_FILLED Only the border of the ellipse is displayed in the color defined in <i>color</i> . Only pixels defined in <i>style</i> are taken into account. FILLED The entire area of the ellipse (including border) is displayed in the color defined in <i>color</i> .
	<i>color</i>	Color: Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
	<i>style</i>	Line style for the ellipse as an 8-bit value. Each bit in the specified numerical value (preferably hexadecimal) represents a pixel. A pixel of the color defined in <i>color</i> is drawn for each set bit. Bits that are not set cause the associated border pixels to remain unchanged. A value of 0xff produces a continuous line.

### 3.1.14 Dynamic ellipse - ELLIPSE\_DYN (MMC100/EBF)

<b>Description</b>	Display an ellipse on the screen; its position, radii and color are read from notepads.	
<b>Syntax</b>	<b>ELLIPSE_DYN</b> ( <i>id, nb_x, nb_y, nb_rx, nb_ry, fill, nb_color, style</i> )	
<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>nb_x, nb_y</i>	Numbers of notepads containing the center point of the ellipse in pixels relative to the reference position (window, menu).



<i>nb_rx, nb_ry</i>	Numbers of notepads containing the radius in x and y directions in pixels.
<i>fill</i>	Number of the fill pattern for the ellipse.  Using <b>DEF_PATTERN</b> (refer below), the user can define the required fill pattern.  Predefined settings are:  NOT_FILLED  Only the border of the dynamic ellipse is displayed in the color defined in <i>color</i> . Only pixels defined in <i>style</i> are taken into account.  FILLED  The entire area of the dynamic ellipse (including border) is displayed in the color defined in <i>color</i> .
<i>nb_color</i>	Number of notepad containing the color. Colors and gray scales are system-dependent (refer to Section <b>Colors</b> for valid values).
<i>style</i>	Line style for the ellipse as an 8-bit value. Each bit in the specified numerical value (preferably hexadecimal) represents a pixel. A pixel of the color defined in <i>color</i> is drawn for each set bit. Bits that are not set cause the associated border pixels to remain unchanged. A value of 0xff produces a continuous line.

### 3.1.15 Fill pattern - definition - DEF\_PATTERN

<b>Description</b>	Using this element, the user can define any fill pattern to fill graphic objects.  A fill pattern consists of a bit pattern, 8 x 8 bits in size, which can be defined by 8 bytes. The first byte represents the top line, i.e. the highest pixel row in the fill pattern. The leftmost pixel on each line is defined by the highest-order bit in a byte.						
<b>Syntax</b>	<b>DEF_PATTERN</b> ( <i>id, pat_nr, pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8</i> )						
<b>Parameters</b>	<table> <tr> <td><i>id</i></td> <td>Element identifier which must be unique within a module for all DEF_PATTERN elements.</td> </tr> <tr> <td><i>pat_nr</i></td> <td>Number of fill pattern. Since 3 fill patterns are assigned to the system and a maximum of 10 can be configured, values between 3 and 9 can be specified here.</td> </tr> <tr> <td><i>pat1...pat8</i></td> <td>8 sequential 8-bit values which represent the pattern, starting with the top line (see above).</td> </tr> </table>	<i>id</i>	Element identifier which must be unique within a module for all DEF_PATTERN elements.	<i>pat_nr</i>	Number of fill pattern. Since 3 fill patterns are assigned to the system and a maximum of 10 can be configured, values between 3 and 9 can be specified here.	<i>pat1...pat8</i>	8 sequential 8-bit values which represent the pattern, starting with the top line (see above).
<i>id</i>	Element identifier which must be unique within a module for all DEF_PATTERN elements.						
<i>pat_nr</i>	Number of fill pattern. Since 3 fill patterns are assigned to the system and a maximum of 10 can be configured, values between 3 and 9 can be specified here.						
<i>pat1...pat8</i>	8 sequential 8-bit values which represent the pattern, starting with the top line (see above).						
<b>Note</b>	If you need more than the 10 fill patterns which can be active simultaneously, you can redefine them within an object list. A new definition for a fill pattern is						

effective only for graphic objects to be displayed subsequently. Elements already displayed on the screen are unaffected. If a graphic object is output with a fill pattern, only the pixels set in the pattern with the color configured in the graphic object are taken into account and displayed.

### 3.1.16 Load color pallet - COL\_TAB (MMC100/EBF)

**Description** You can load any color pallet you have defined yourself using this element. The color pallet definition is stored in an ASCII file (you specify the file name), which is structured as follows (s syscol.cnf):

```
;      Red      Green      Blue
      10,      20,      30      ; color 0
```

All characters after ; are interpreted as comment text. You can define up to 16 colors. You can set the color intensity to values between 0 and 63. The color produced in the example above consists of 10 parts red, 20 parts green and 30 parts blue.

**Syntax** COL\_TAB (*id, file*)

**Parameters**

<i>id</i>	Element identifier which must be unique within a module for all DEF_PATTERN elements.
<i>file</i>	Name of ASCII file containing the pallet definition (max. 14 characters including extension).

**Note** You can link this file into the target system using the "Copy external Files into Project..." instruction in the installation kit.

### 3.1.17 Texts - TEXT, FIXTEXT

**Description** 2 objects are provided for displaying texts on the screen.

#### 1) TEXT

On one hand, text can be output that are centrally saved in configurable text lists. The list element **TEXT** is used for this purpose. The texts are represented by using text numbers, whereby each text is assigned a unique text number (*txt\_nr*) that is then used in the object list for the configuring.

#### 2) FIXTEXT

The second option employs the **FIXTEXT** list element with which fixed texts can be output. In this case, it is not a text number for accessing a text list that is

transferred, but a fixed text. The text itself must be in " ", and must not be longer than 49 characters including this identifier

---

**Note**

FIXTEXT(s) must be language-independent.

---

**Syntax**

For texts from text lists:

**TEXT** (*txt\_id*, *x*, *y*, *txt\_nr*, *char\_set*, *attr*[*attr*], *color*)

For predefined texts:

**FIXTEXT** (*ftxt\_id*, *x*, *y*, *text*, *char\_set*, *attr*[*attr*], *color*)

**Parameters**

<i>txt_id</i> , <i>ftxt_id</i>	Unique identifier of element.
<i>x</i>	<i>x</i> position of text in pixels relative to the higher-level object. The reference point in the <i>x</i> direction depends on the attribute ( <i>attr</i> ).
<i>y</i>	<i>y</i> position of upper edge of character matrix in pixels relative to the higher-level object.
<i>txt_nr</i>	Number of text to be displayed from the text list.
<i>text</i>	Text to be displayed (max. 49 characters, text is contained within " ").
<i>char_set</i>	Identifier for character set to be used. CS_SMALL Character set 8 x 12 pixels ( <b>not HT6</b> ) Character set 6 x 8 pixels ( <b>for HPU</b> ) CS_6_9: Character set 6 x 9 pixels (for MMC 100/UOP only); character set must be loaded first using AC/RC_LD_FONT. CS_8_12: (as for CS_SMALL; <b>only for MMC 100/UOP</b> ) CS_19_27: Character set 19 x 27 pixels ( <b>only for MMC 100/UOP</b> ); character set must be loaded first using AC/RC_LD_FONT.
<i>attr</i>	Attribute word; individual attributes can be combined. TEXT_DOUBLE_HEIGHT The characters are output with twice the height of the specified character set. TEXT_DOUBLE_ZOOMED

The characters are doubled in the x and y directions and thus displayed with double line thickness.

**TEXT\_WRITE\_LF**

Line feeds in the display text are represented symbolically.

**TEXT\_RIGHT\_ADJUST**

The text is right justified, i.e. written to the left from the specific x position. (Left-justified is the default setting.)

**TEXT\_CENTRE\_ADJUST**

The text is centered, i.e. the configured x position represents the center of the display text.

**TEXT\_PIXEL\_ADJUST** (not OP 030, HPU)

The text is output at the exact pixel position. If this attribute is not set, the text is output at the next-lower x position divisible by 8 (byte limit) to save time.

0

The text is left-justified. To save time, the text is displayed at the next-lower x position divisible by 8 (byte limit).

*color* Color: Colors and gray scales are system-dependent (refer to Section Colors for valid values).

**Note**

A text enclosed in quotation marks (" ") is interpreted as a text pointer.

OP030 MMC 100/UOP to SW4

To link language-specific special characters, e.g. ä, ö, ü into the text, you must create it using an OEM editor (DOS editor e.g. edit)

MMC 100/UOP, SW5 and later:

To integrate language-specific special characters, you must now use an ANSI or a Windows editor (Visual C Workbench, Notepad).

**Special features**

Text strings can contain the following control characters

- %n** Line break
- %#** Insert the text number
- %@x** Insert the axis name of the x axis  
(only for MMC 100 /UOP, SW version 4.0 and later)
- %px** Insert plane-dependent axis name of the xth axis. The plane is read from notepad NB\_SY\_PLANE  
(only for MMC 100/UOP, SW version 4.0 and later)
- %ix** Insert the plane-dependent xth circle parameter. The plane is read from notepad NB\_SY\_PLANE (only for MMC 100/UOP, SW version 4.0 and later)

Content of NB_SY_PLANE	Control character: %p1	Control character: %p2	Control character: %p3
1 (X-Y plane)	Axis name axis 1	Axis name axis 2	Axis name axis 3
2 (Z-X plane)	Axis name axis 3	Axis name axis 1	Axis name axis 2
3 (Y-Z plane)	Axis name axis 2	Axis name axis 3	Axis name axis 1

Content of NB_SY_PLANE	Control character: %i1	Control character: %i2	Control character: %i3
1 (X-Y plane)	I	J	K
2 (Z-X plane)	K	I	J
3 (Y-Z plane)	J	K	I

#### Special characters:


In SW version 4.4. and later, a Siemens standard symbol font is available which allows special characters to be displayed independently of language.

%<font name,font index> is defined as identifier.














The Siemens symbol font is named, for example, "sym1". The font index refers to the respective entry in the file.

#### Example:






















Text file : "You can display special character %<Sym1,23> like this"

Display : You can display special character  like this.

#### Special characters from the Siemens standard symbol file:

Font index	Character	Description	Application
1	„	Inverted comma	%<Sym1,1>
2		Tool position identifier	%<Sym1,2>
3		Tool position identifier	%<Sym1,3>
4		Tool position identifier	%<Sym1,4>
5		Tool position identifier	%<Sym1,5>
6		Tool position identifier	%<Sym1,6>
7		Tool position identifier	%<Sym1,7>
8		Tool position identifier	%<Sym1,8>
9		Tool position identifier	%<Sym1,9>
10		Tool position identifier	%<Sym1,10>
11		Roughing	%<Sym1,11>
12		alpha	%<Sym1,12>
13		CW rotation	%<Sym1,13>
14		CCW rotation	%<Sym1,14>

## 3.1 Static display elements

15		Diameter	%<Sym1,15>
16		1. Part 3 Position spindle	%<Sym1,16>%<Sym1,17>%<Sym1,18>
17		2. Part 3 Position spindle	%<Sym1,16>%<Sym1,17>%<Sym1,18>
18		3. Part 3 Position spindle	%<Sym1,16>%<Sym1,17>%<Sym1,18>
19		Degrees celsius	%<Sym1,19>
20		beta	%<Sym1,20>
21		Arrow down	%<Sym1,21>
22		Delta	%<Sym1,22>
23		Spindle	%<Sym1,23>
24		Cooling water	%<Sym1,24>
25		Spindle stop	%<Sym1,25>
26		1. Part 2 Machine key	%<Sym1,27>%<Sym1,27>
27		2. Part 2 Machine key	%<Sym1,27>%<Sym1,27>
28		1. Part 2 Tool nose radius compensation to left of contour	%<Sym1,28>%<Sym1,29>
29		2. Part 2 Tool nose radius compensation to left of contour	%<Sym1,28>%<Sym1,29>
30		1. Part 2 Tool nose radius compensation to right of contour	%<Sym1,30>%<Sym1,31>
31		2. Part 2 Tool nose radius compensation to right of contour	%<Sym1,30>%<Sym1,31>
32		1. Part 2 Tool nose radius compensation Off	%<Sym1,32>%<Sym1,33>
33		2. Part 2 Tool nose radius compensation Off	%<Sym1,32>%<Sym1,33>
34		Gripper right	%<Sym1,34>
35		Gripper left	%<Sym1,35>

## 3.1.18 Dynamic texts - TEXT\_DYN

## Description

Display text on the screen; its position, text identification and color are read from notepads.

## Syntax

**TEXT\_DYN** (*id, nb\_x, nb\_y, nb\_txt\_nr, char\_set, attr[*attr*], nb\_color*)

<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>nb_x</i>	Number of notepad containing the X position of the text in pixels relative to the higher-level object. The reference point in the x direction depends on the attribute ( <i>attr</i> ).
	<i>nb_y</i>	Number of notepad containing the Y position of the upper edge of the character matrix in pixels relative to the higher-level object.
	<i>nb_txt_nr</i>	Number of notepad containing the ID number of the display text from the text list.
	<i>char_set</i>	Identifier for character set to be used. CS_SMALL Character set 8 x 12 pixels ( <b>not HT6</b> ) Character set 6 x 8 pixels ( <b>for HPU</b> ) CS_6_9: Character set 6 x 9 pixels (for MMC 100/UOP only); character set must be loaded first using AC/RC_LD_FONT. CS_8_12: (as for CS_SMALL; <b>only for MMC 100/UOP</b> ) CS_19_27: Character set 19 x 27 pixels ( <b>only for MMC 100/UOP</b> ); character set must be loaded first using AC/RC_LD_FONT.
	<i>attr</i>	Attribute word; individual attributes can be combined. TEXT_DOUBLE_HEIGHT The characters are output with twice the height of the specified character set. TEXT_DOUBLE_ZOOMED The characters are doubled in the x and y directions and thus displayed with double line thickness. TEXT_WRITE_LF Linefeeds in the display text are represented symbolically. TEXT_RIGHT_ADJUST The text is right-adjusted, i.e. written to the right from the specific x position. (Left-justified is the default setting.) TEXT_CENTRE_ADJUST The text is centered, i.e. the configured x position represents the center of the display text. TEXT_PIXEL_ADJUST (not OP 030, HPU)

The text is output at the exact pixel position. If this attribute is not set, the text is output at the next-lower x position divisible by 8 (byte limit) to save time.

*nb\_color* Number of notepad containing the color. Colors and gray scales are system-dependent (refer to the Section **Colors** for valid values).

**Note** See section "Texts – TEXT, FIXTEXT"

### 3.1.19 Polymarker definition - DEF\_POLYMARKER

#### Description

You can define any number of polymarkers using this element.

A polymarker consists of a bit pattern, 16 x 16 pixels in size, which can be defined by 16 data words (2 bytes each). The first data word represents the top line or pixel row in the bit pattern. The left-most pixel on each line is defined by the highest-order bit in a word.

A polymarker is identified by its number. The assignment between pattern and number remains valid until a new polymarker is defined for the number.

#### Syntax

**DEF\_POLYMARKER** (*id, poly\_nr, p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16*)

#### Parameters

<i>id</i>	Element identifier which must be unique within a module for all DEF_POLYMARKER elements.
<i>poly_nr</i>	Number of polymarker. The numbers 0 to 99 are reserved by the system. Numbers 100 to 199 are available to users.
<i>p1 ...p16</i>	16 unsigned words, where the 1st word (p1) defines the top line of the polymarker and the 16th word (p16) the bottom line.



### 3.1.20 Polymarker definition - DEF\_POLYMARKER (only MMC100/EBF)

<b>Description</b>	<p>You can define any number of polymarkers within an application using this element.</p> <p>A polymarker consists of a bit pattern, 16 x 16 pixels in size, which can be defined by 16 data words (2 bytes each). The first data word represents the top line or pixel row in the bit pattern. The left-most pixel on each line is defined by the highest-order bit in a word.</p> <p>A polymarker is identified by its number.</p> <p>Polymarker numbering starts at 200 (LOCAL_POLY_LIST) for every application. The second polymarker is numbered 201, etc., gaps cannot be left in the number sequence.</p> <p>Only one polymarker list can be created per application.</p> <p>The polymarker list must be registered with the system in the system directory (ap_l_dir.h) with</p> <pre>/* external declaration for polymarker (APP_MENU.C) */ EXTERN_POLY_LIST(LOCAL_POLY_LIST,name).</pre>		
<b>Syntax</b>	<pre><b>BEGIN_POLY_LIST(LOCAL_POLY_LIST, name)</b> /* 16 data words for describing polymarker 200*/ {0x0000, 0x0000, 0x01f0, 0x0110, /* 200 */ 0x3f1c, 0x0114, 0x0114, 0x0114, 0x0114, 0x0114, 0x0114, 0x3f1c, 0x0110, 0x01f0, 0x0000, 0x0000}, /* 16 data words for describing polymarker 201*/ {0x0000, 0x0000, 0x01f0, 0x0110, /* 201 */ 0x3f1c, 0x0114, 0x0114, 0x0114, 0x0114, 0x0114, 0x0114, 0x3f1c, 0x0110, 0x01f0, 0x0000, 0x0000},  {...} <b>END_POLY_LIST(LOCAL_POLY_LIST, name)</b></pre>		
<b>Parameters</b>	<table border="0"> <tr> <td style="padding-right: 20px;">name</td> <td>Identifier of polymarker list which must be unique throughout the whole system. Entry of the application name here is recommended, e.g. example.</td> </tr> </table>	name	Identifier of polymarker list which must be unique throughout the whole system. Entry of the application name here is recommended, e.g. example.
name	Identifier of polymarker list which must be unique throughout the whole system. Entry of the application name here is recommended, e.g. example.		
<b>Note</b>	We recommend that you structure the polymarker numbering for subsequent accessing with configuring macro POLYMARKER as illustrated below:		

```
#define MY_FIRST_POLYMARKER      LOCAL_POLY_LIST
#define MY_SECOND_POLYMARKER     MY_FIRST_POLY_MARKER+1
#define MY_THIRD_POLYMARKER      MY_FIRST_POLY_MARKER+2
etc.
```

**Example**

Add polymarkers to the sample application in the screen kit.

Entry in file ap\_l\_dir.h:

```
/* external declaration for polymarker (APP_MENU.C) */
```

```
EXTERN_POLY_LIST(LOCAL_POLY_LIST, example)
```

app\_menu.c:

```
/*-----
 *   © Siemens AG, 1996. All rights reserved Confidential
 *-----
 * Module:      APP_MENU.C
 * Name:        created by Screen Kit
 *-----
 * Description:  Basic sample for Application
 *-----
 */
#include "proj.h"
#include "mwl_app.h"
#include "example.h"
#include "app_incl.h"
#include "nb_app.h"

/* points to the used macros defined in APP_CTRL.C */
EXTERN_OBJECT_LIST (OB_CLEAR_WINDOW)
EXTERN_OBJECT_LIST (SOB_CLEAR_HOR_SK)
EXTERN_OBJECT_LIST (SOB_CLEAR_VER_SK)

#define MY_FIRST_POLY      LOCAL_POLY_LIST      /*200*/
#define MY_SECOND_POLY     MY_FIRST_POLY+1      /*201*/
#define MY_THIRD_POLY      MY_FIRST_POLY+2      /*202*/
#define MY_FOURTH_POLY     MY_FIRST_POLY+3      /*203*/
```

**BEGIN\_POLY\_LIST(LOCAL\_POLY\_LIST, example)**

```

/* Spindle stopped */
{ 0x0000, 0x0000, 0x01f0, 0x0110, /* 200 */
  0x3f1c, 0x0114, 0x0114, 0x0114,
  0x0114, 0x0114, 0x0114, 0x3f1c,
  0x0110, 0x01f0, 0x0000, 0x0000},

/* manual */
{ 0x0000, 0x1ff0, 0x22a8, 0x4154, /* 201 */
  0x98aa, 0xa401, 0x4200, 0x0100,
  0x1f00, 0x2000, 0x2000, 0x1ff8,
  0x0004, 0x0002, 0x0000, 0x0000},

/* Reference point */
{ 0x0180, 0x0180, 0x07E0, 0x09F0, /* 202 */
  0x11F8, 0x21FC, 0x21FC, 0xFFFF,
  0xFFFF, 0x3F84, 0x3F84, 0x1F88,
  0x0F90, 0x07E0, 0x0180, 0x0180},

/* Work offset */
{ 0x2000, 0x7040, 0xA8E0, 0x2040, /* 203 */
  0x2040, 0x2044, 0x207E, 0x2384,
  0x2180, 0x2280, 0x2400, 0x2804,
  0x3002, 0x3FFF, 0x0002, 0x0004},

```

**END\_POLY\_LIST(LOCAL\_POLY\_LIST, example)**

```

BEGIN_OBJECT_LIST (W_WIN_1)
/* clearing window: macro defined in APP_CTRL.C */
MACRO (110, 0, 0, OBJECT_LIST_PTR (OB_CLEAR_WINDOW))

/* printing window-header_text */
TEXT (120, HEADER_X, HEADER_Y, T_HEADER_TEXT_1, CS_SMALL, 0,
W_TCOL)

/* printing text comment for R parameter(10) */
TEXT (130, X_T_APP + 100, (Y_T_APP + (0 * Y_T_DIFF)), T_R_PARAM_10,
CS_SMALL, 0, W_TCOL)
/* visualizing R10 */

```

## 3.1 Static display elements

```

IO_FIELD (140, X_T_APP, (Y_T_APP + (0 * Y_T_DIFF)), 10,
          W_O_TCOL, W_O_FCOL,
          CS_SMALL,
          0,
          160, 160, 160, 160,
          0,
          1,
          0,
          P_C_RP_rpa, 10+1, 0, 0,
          CON_DECIMAL, F_DOUBLE, 8, 0)

```

```

/* printing text comment for NB20 */
TEXT (150, X_T_APP + 100, (Y_T_APP + (1 * Y_T_DIFF)), T_NB20,
      CS_SMALL, 0, W_TCOL)
/* visualizing NB20 */
IO_FIELD (160, X_T_APP, (Y_T_APP + (1 * Y_T_DIFF)), 10,
          W_O_TCOL, W_O_FCOL,
          CS_SMALL,
          0,
          140, 140, 140, 140,
          0,
          1,
          T_COMMENT_NB20,
          P_NB, NB_CALC_WITH_R10, 0, 0,
          CON_DECIMAL, F_DOUBLE, 8, 0)

```

```

FIXTEXT (100, 20, 100, "Polymarker:", CS_SMALL, 0, W_TCOL )

```

```

POLYMARKER(101, 50, 130, MY_FIRST_POLY, W_TCOL)
POLYMARKER(102, 80, 130, MY_SECOND_POLY, W_TCOL)
POLYMARKER(103, 110, 130, MY_THIRD_POLY, W_TCOL)
POLYMARKER(104, 140, 130, MY_FOURTH_POLY, W_TCOL)

```

```

END_OBJECT_LIST (W_WIN_1)

```

```

BEGIN_SOFTKEY_REACTION_LIST (W_WIN_1)
/*+-----+*/
/*|KEY_F1_V|*/
/*+-----+*/
/* inverting first vertical soft key */
RC_DRAW_SOFTKEY (200, KEY_F1_V, T_SK1V_ADD, KEY_F1_V, 2,
                 PRESSED)
/* add 1.234 to R parameter 10 */
RC_CALC_DATA (210, KEY_F1_V,
              P_C_RP_rpa, 10+1, 0, 0,
              AC_ADD,
              P_NB, NB_CALC_WITH_R10, 0, 0, BTSS_DOUBLE)
/* re-inverting first vertical soft key */
RC_DRAW_SOFTKEY (220, KEY_F1_V, T_SK1V_ADD, KEY_F1_V, 2,
                 NOT_PRESSED)
/*+-----+*/
/*|KEY_F2_V|*/
/*+-----+*/
/* inverting second vertical soft key */
RC_DRAW_SOFTKEY (230, KEY_F2_V, T_SK2V_SUB, KEY_F2_V, 2,

```

```

PRESSED)
/* subtract 1.234 from R-parameter 10 */
RC_CALC_DATA (240, KEY_F2_V,
              P_C_RP_rpa, 10+1, 0, 0,
              AC_SUB,
              P_NB, NB_CALC_WITH_R10, 0, 0, BTSS_DOUBLE)
/* reinverting second vertical soft key */
RC_DRAW_SOFTKEY (250, KEY_F2_V, T_SK2V_SUB, KEY_F2_V, 2,
NOT_PRESSED)
/*+-----+*/
/*|KEY_F1 |*/
/*+-----+*/
/* close win 1 */
RC_CLOSE_WINDOW (270, KEY_F1, W_WIN_1, LOCAL, 1)
/* open win 2 */
RC_OPEN_WINDOW (280, KEY_F1, W_WIN_2, LOCAL)
END_SOFTKEY_REACTION_LIST (W_WIN_1)

BEGIN_SOFTKEY_OBJECT_LIST (W_WIN_1)
/* clearing soft keys: macros defined in APP_CTRL.C */
MACRO (300, 0, 0, OBJECT_LIST_PTR (SOB_CLEAR_HOR_SK))
MACRO (310, 0, 0, OBJECT_LIST_PTR (SOB_CLEAR_VER_SK))
/* printing soft key texts */
SOFTKEY (320, T_SK1V_ADD, KEY_F1_V, 2, NOT_PRESSED)
SOFTKEY (330, T_SK2V_SUB, KEY_F2_V, 2, NOT_PRESSED)
SOFTKEY (340, T_SK1H_CHANGE_TO_WIN2, KEY_F1, 2, NOT_PRESSED)
END_SOFTKEY_OBJECT_LIST (W_WIN_1)

BEGIN_OPEN_LIST (W_WIN_1)
/* init R10 with 123.456789 */
AC_SET_DOUBLE (400, 123.456789, P_C_RP_rpa, 10+1, 0, 0)
/* init NB20 with 987.654321 */
AC_SET_DOUBLE (410, 987.654321, P_NB, NB_CALC_WITH_R10, 0, 0)
END_OPEN_LIST (W_WIN_1)

BEGIN_WINDOW (W_WIN_1)
0,
0, 0,
WIDTH_M_INI, HEIGHT_M_INI,
W_BCOL,
W_FCOL,
NULL,
OPEN_LIST_PTR (W_WIN_1),
NULL,
OBJECT_LIST_PTR (W_WIN_1),
NULL,
SOFTKEY_OBJECT_LIST_PTR (W_WIN_1),
SOFTKEY_REACTION_LIST_PTR (W_WIN_1)
END_WINDOW (W_WIN_1)

BEGIN_MENU (M_MENU_1)
M_CLEAR_BACKGROUND,
0,
W_WIN_1,
X_M_INI, Y_M_INI,
WIDTH_M_INI, HEIGHT_M_INI,
BK_CL_FCOL,
NULL,
NULL
END_MENU (M_MENU_1)

```

### 3.1.21 Polymarker - POLYMARKER

<b>Description</b>	Displays a polymarker (Icon with 16 * 16 pixels) that has been defined with DEF_POLYMARKER on the screen.	
<b>Syntax</b>	<b>POLYMARKER</b> ( <i>id, x, y, poly_nr, color</i> )	
<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>x, y</i>	Position of top left-hand corner of the polymarker in pixels relative to the reference position (window, menu).
	<i>poly_nr</i>	Number of polymarker. 0 to 99 : Reserved for SIEMENS 100 to 199 : Polymarker defined by user
	<i>color</i>	Color: Colors and gray stages depend on the system (refer to the Section – <b>Colors</b> for valid values).
<b>Note</b>	Polymarkers have no background color, i.e. only the pixels to be set are output on the screen, all other areas are left in their original state.  To obtain a defined background color for a polymarker, a rectangle with the desired color must be output in the polymarker position beforehand.	

### 3.1.22 Dynamic polymarker - POLYMARKER\_DYN (MMC100/EBF)

<b>Description</b>	Displays a polymarker (icon with 16 * 16 pixels) that has been defined with DEF_POLYMARKER on the screen. Its position, identity and color are read from notepads.	
<b>Syntax</b>	<b>POLYMARKER_DYN</b> ( <i>id, nb_x, nb_y, nb_poly_nr, nb_color</i> )	
<b>Parameters</b>	<i>id</i>	Unique identifier of element.
	<i>nb_x, nb_y</i>	Notepads containing the position of the top, left-hand corner of the polymarker in pixels relative to the reference position (window, menu).
	<i>nb_poly_nr</i>	Number of notepad containing the number of the polymarker. The following applies to polymarker numbers: 0 to 99 : Reserved for SIEMENS 100 to 199 : Polymarker defined by user

*nb\_color*            Number of notepad containing the color. Colors and gray scales are system-dependent (refer to the Section **Colors** for valid values).

**Note**                    Polymarkers have no background color, i.e. only the pixels to be set are output on the screen, all other areas are left in their original state.

To obtain a defined background color for a polymarker, a rectangle with the desired color must be output in the polymarker position beforehand.

### 3.1.23 SOFT KEY

**The following applies to the OP 030 and MMC 100/UOP**

#### Description

Display a soft key on the screen.

This list element may only be used in a soft key object list.

#### Syntax

**SOFTKEY** (*id, sk\_txt\_nr, sk, sk\_lines, sk\_sts*)

#### Parameters

<i>id</i>	Unique identifier of element.
<i>sk_txt_nr</i>	Number of text to be displayed from a text list.
<i>sk</i>	Identifier / designation of the soft key to be output. OP 030: KEY_F1                    KEY_F5 MMC 100/UOP: KEY_F1            KEY_F8 =>horizontal soft key KEY_F1_V            KEY_F8_V =>vertical soft key
<i>sk_lines</i>	Number of text lines of the soft key (for OP 030, always 2 ).
<i>sk_sts</i>	Status of the soft keys. PRESSED, SK_PRESSED The soft key is displayed in reverse video. NOT_PRESSED, SK_NOT_PRESSED The soft key is displayed normally. SK_DEACTIVATED The soft key is displayed with semi-concealed print on the screen. It is deactivated.

#### Note

Soft key displays are **not** supported by a system function on the handheld programming unit (HPU).

The list language described in this document is used to display soft keys on the screen, thereby enabling the configuring engineer to directly influence the mode of representation.

## 3.1 Static display elements

The standard configuration resides in files

\proj\_hpu\std\src\softkey.c and \proj\_hpu\h\softk.h

allows a representation possibility using macros, that can also be used in the application. These macros are structured in the same form as the list language.

**The following applies only to the HPU**

**Description**                      Initializes the soft key line or displays an empty soft key line incl. representation for active "large function keys".

**Syntax**                              **INIT\_SK\_LINE** (*id*)

**Parameters**                      *id*                              Unique identifier of element.

**Description**                      Displays a soft key text in reverse video for soft key x on the screen(x = 1 to 5).

**Syntax**                              **SKx\_PRESSED** (*id, sk\_txt\_nr*)

**Parameters**                      *id*                              Unique identifier of element.  
*sk\_txt\_nr*                      Number of text to be displayed from a text list.

**Description**                      Displays a soft key text for soft key x on the screen (x = 1 to 5).  
This list element may only be used in a soft key object list.

**Syntax**                              **SKx\_TEXT** (*id, sk\_txt\_nr*)

**Parameters**                      *id*                              Unique identifier of element.  
*sk\_txt\_nr*                      Number of text to be displayed from a text list.

**Example**                              BEGIN\_SOFTKEY\_OBJECT\_LIST (SOB\_W\_CIRC\_DP)  
  INIT\_SK\_LINE (101)  
  SK1\_TEXT                      (2304, T\_PS\_SK\_G00)  
  SK2\_TEXT                      (2305, T\_PS\_SK\_G01)  
  SK3\_PRESSED                (2310, T\_PS\_SK\_CIP)  
  SK4\_TEXT                      (2315, T\_PS\_SK\_VP)  
  SK5\_TEXT                      (2320, T\_PS\_SK\_AXES)  
  END\_SOFTKEY\_OBJECT\_LIST (SOB\_W\_CIRC\_DP)



### 3.1.24 SOFTKEY\_PRO

<b>Description</b>	<p>This element represents a soft key with expanded parameterizability. It can only be used in object lists and soft key object lists.</p> <p>This element allows either 2 soft key texts or 2 bitmaps to be configured within a soft key or one text and one bitmap. The representation can be configured so that the two elements can be written above one another, the soft key can either be halved vertically or horizontally. Further, it is also possible to freely position both elements – graphics or text – within the soft key.</p> <p>The text, that starts with "\\\" is always interpreted as a name of a bitmap file.</p>												
<b>Syntax</b>	<b>SOFTKEY_PRO</b> ( <i>id, sk, tid1, tid2, active_attr, layout_attr, x1, y1, x2, y2</i> )												
<b>Parameters</b>	<table border="0"> <tr> <td style="vertical-align: top;"><i>id</i></td> <td>Unique identity within the C module in which the progress bar was defined.</td> </tr> <tr> <td style="vertical-align: top;"><i>sk</i></td> <td>Number of the soft key (KEY_F1...KEY_F8V)</td> </tr> <tr> <td style="vertical-align: top;"><i>tid1</i></td> <td>1. Text identity (this text is first displayed).</td> </tr> <tr> <td style="vertical-align: top;"><i>tid2</i></td> <td>2. Text identity (this part is displayed as second).</td> </tr> <tr> <td style="vertical-align: top;"><i>active_attr</i></td> <td>Soft key attribute:  <b>SK_NOT_PRESSED</b>  Normal soft key representation   <b>SK_PRESSED</b>  Active soft key (colors are inverted)   SK_DEACTIVATED  Inactive soft key (gray text) </td> </tr> <tr> <td style="vertical-align: top;"><i>layout_attr</i></td> <td>Attributed for the distribution of the text/graphic elements on the soft key. The attribute <b>cannot</b> be OR'ed bitwise:   SK_LAYOUT_NO_SPLIT  Both elements are displayed at the top left on the soft key at the same position (standard)   SK_LAYOUT_HORIZONTAL_SPLIT  Soft key is split horizontally at the center. tid1 is displayed in the upper half and tdi2 in the lower half of the soft key.   SK_LAYOUT_VERTICAL_SPLIT  Soft key is split vertically at the center. tid1 is displayed in the left half and tdi2 in the right half of the soft key.   SK_LAYOUT_POSITION  Both elements can be freely position within the soft key. The following position parameters are only effective if this attribute is set. </td> </tr> </table>	<i>id</i>	Unique identity within the C module in which the progress bar was defined.	<i>sk</i>	Number of the soft key (KEY_F1...KEY_F8V)	<i>tid1</i>	1. Text identity (this text is first displayed).	<i>tid2</i>	2. Text identity (this part is displayed as second).	<i>active_attr</i>	Soft key attribute: <b>SK_NOT_PRESSED</b> Normal soft key representation  <b>SK_PRESSED</b> Active soft key (colors are inverted)  SK_DEACTIVATED Inactive soft key (gray text)	<i>layout_attr</i>	Attributed for the distribution of the text/graphic elements on the soft key. The attribute <b>cannot</b> be OR'ed bitwise:  SK_LAYOUT_NO_SPLIT Both elements are displayed at the top left on the soft key at the same position (standard)  SK_LAYOUT_HORIZONTAL_SPLIT Soft key is split horizontally at the center. tid1 is displayed in the upper half and tdi2 in the lower half of the soft key.  SK_LAYOUT_VERTICAL_SPLIT Soft key is split vertically at the center. tid1 is displayed in the left half and tdi2 in the right half of the soft key.  SK_LAYOUT_POSITION Both elements can be freely position within the soft key. The following position parameters are only effective if this attribute is set.
<i>id</i>	Unique identity within the C module in which the progress bar was defined.												
<i>sk</i>	Number of the soft key (KEY_F1...KEY_F8V)												
<i>tid1</i>	1. Text identity (this text is first displayed).												
<i>tid2</i>	2. Text identity (this part is displayed as second).												
<i>active_attr</i>	Soft key attribute: <b>SK_NOT_PRESSED</b> Normal soft key representation  <b>SK_PRESSED</b> Active soft key (colors are inverted)  SK_DEACTIVATED Inactive soft key (gray text)												
<i>layout_attr</i>	Attributed for the distribution of the text/graphic elements on the soft key. The attribute <b>cannot</b> be OR'ed bitwise:  SK_LAYOUT_NO_SPLIT Both elements are displayed at the top left on the soft key at the same position (standard)  SK_LAYOUT_HORIZONTAL_SPLIT Soft key is split horizontally at the center. tid1 is displayed in the upper half and tdi2 in the lower half of the soft key.  SK_LAYOUT_VERTICAL_SPLIT Soft key is split vertically at the center. tid1 is displayed in the left half and tdi2 in the right half of the soft key.  SK_LAYOUT_POSITION Both elements can be freely position within the soft key. The following position parameters are only effective if this attribute is set.												

3.1 Static display elements

- $x1, y1$  Free positions for the 1st element (this is only effective if the attribute bit SK\_LAYOUT\_POSITION is set in the parameter LAYOUT\_ATTR). This element is not displayed if the position is invalid.
- $x2, y2$  Free positions for the 2nd element (this is only effective if the attribute bit SK\_LAYOUT\_POSITION is set in the parameter LAYOUT\_ATTR). This element is not displayed if the position is invalid.

Layout attribute **SK\_LAYOUT\_NO\_SPLIT:**



- 1st soft key: Only the 1st element has text
- 2nd soft key: Only the 1st element has a bitmap
- 3rd soft key: 1st element has a bitmap, 2nd element has text
- 4th soft key: Only the 1st element has a bitmap, attribute SK\_PRESSED set
- 5th soft key: Only the 2nd element has a bitmap, attribute SK\_PRESSED set
- 6th soft key: 1st element has a bitmap, 2nd element has text, attribute SK\_PRESSED set.
- 7th soft key: Only the 2nd element has text, attribute SK\_DEACTIVATED set

Layout attribute **SK\_LAYOUT\_HORIZONTAL\_SPLIT:**



- 1st soft key: Both elements have the same text
- 2nd soft key: Both elements have the same bitmap
- 3rd soft key: 1st element has a bitmap, 2nd element has text
- 4th soft key: 1st element has text, 2nd element has a bitmap
- 5th soft key: as for soft key 1, however attribute SK\_PRESSED set
- 6th soft key: as for soft key 2, however attribute SK\_PRESSED set
- 7th soft key: as for soft key 3, however attribute SK\_PRESSED set
- 8th soft key: as for soft key 4, however attribute SK\_PRESSED set

Layout attribute **SK\_LAYOUT\_VERTICAL\_SPLIT:**



- 1st soft key: Both elements have the same text
- 2nd soft key: Both elements have the same bitmap
- 3rd soft key: 1st element has a bitmap, 2nd element has text
- 4th soft key: 1st element has text, 2nd element has a bitmap
- 5th soft key: as for soft key 1, however attribute SK\_PRESSED set
- 6th soft key: as for soft key 2, however attribute SK\_PRESSED set
- 7th soft key: as for soft key 3, however attribute SK\_PRESSED set

8th soft key: as for soft key 4, however attribute SK\_PRESSED set

## 3.2 Dynamic display elements – dialog fields

<b>Description</b>	Data of the NCK, PLC and MMC can be displayed by means of dynamic display elements and modified through user inputs. The following configurable list elements can be programmed in window object lists:
	Input/output field (IO_FIELD)
	Output field only (O_FIELD)
	Input field only (I_FIELD)
	Editor/NC data overview (EDIT_FIELD)
	Table element (TABLE, TAB_FIELD, TAB_ITEM)
	Icons (PICT_FIELD)
	Single/multiple check field (CHECK_FIELD)
	Action field (ACTION_FIELD)
	Reverse video field (INVERSE_FIELD)
	Scrollbar (DEF_SCROLL_BAR)
	Macro – sub-object lists (MAKRO)
	Progress bar (PROGRESS_BAR)
	Tacho element (TACHO)

These objects are displayed in the windows assigned by the window object list and window definition block.

### 3.2.1 Input/output field - IO\_FIELD

<b>Description</b>	The contents of NC variables, PLC variables, MMC variables and MMC methods are made accessible on the operator interface via an input/output field. The contents are displayed in various modes depending on parameter settings. You can also enter values in them as a function of the parameter settings. You can specify the refresh cycle and the sequence in which the fields are processed by setting individual parameters.
--------------------	---

<b>Syntax</b>	<b>IO_FIELD</b> ( <i>id, x, y, w, fc, bc, char_set, field_attr[field_attr...] [access class], cur_r, cur_l, cur_d, cur_u, acc_class, refresh,</i>
---------------	---

*cursor\_txt\_id,*  
*v\_adr, v\_p1, v\_p2, v\_p3,*  
*con, con\_p1, con\_p2, con\_p3)*

<b>Parameters</b>	<i>id</i>	Unique identifier for the IO_FIELD object in the current module.
	<i>x, y</i>	Position of top left-hand corner of field in pixels relative to the object above it.
	<i>w</i>	Field width in <b>number of characters</b> . This parameter, together with the selected character set and the text attribute, determines the width and height of the field.
	<i>fc</i>	Foreground color (text color). Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
	<i>bc</i>	Background color. (You can select foreground and background color in the same color to implement keyword fields.) Colors and gray scales are system-independent (see Section <b>Colors</b> for valid values).
	<i>char_set</i>	Identifier for character set to be used.  CS_SMALL Character set 8 x 12 pixels ( <b>not HT6</b> ) Character set 6 x 8 pixels ( <b>for HPU</b> )  CS_6_9: Character set 6 x 9 pixels (for MMC 100/UOP only); character set must be loaded first using AC/RC_LD_FONT.  It must be ensured in the installation kit that the font file is included in the system (with ...Copy internal Files into Project).  CS_8_12: (as for CS_SMALL; <b>only for MMC 100/UOP</b> )  CS_19_27: Character set 19 x 27 pixels ( <b>only for MMC 100/UOP</b> ); character set must be loaded first using AC/RC_LD_FONT.
	<i>field_attr</i>	Attribute parameter. Individual attributes can be combined:  TEXT_DOUBLE_HEIGHT Characters are displayed in double the height defined in the appropriate selected character set.  TEXT_DOUBLE_ZOOMED The characters are doubled in the x and y directions and thus displayed with double line thickness.  TEXT_WRITE_LF Line feed is output as characters "LF".

**IO\_LEFT\_ADJUST**

Text is left-justified within the field.

**IO\_CENTRE\_ADJUST**

Text is centered within the field.

If you do not specify either of the two justification attributes (IO\_LEFT\_ADJUST, IO\_CENTRE\_ADJUST), the following standard applies: Text is left-justified in spring fields (conversion format CON\_STRING or CON\_STRING\_LIMIT) and right-justified with value formats (all other formats including CON\_ASCII). Only **one** of the two justification attributes may be combined with the attribute word.

**IO\_CALCULATOR\_OFF**

No pocket calculator function if the input represents a numerical value. (For use of pocket calculator function, see Operator's Guide)

**IO\_EDITOR\_SCROLL**

Allows you to enter more characters than defined in parameter *w*. The input is then scrolled within the field.

**IO\_INPUT\_DISABLE**

You can select the field with the dialog cursor, but you cannot open it for editing.

**IO\_CURSOR\_DISABLE**

Field is skipped by the dialog cursor.

**DIA\_CURSOR\_STOP**

The cursor remains on this field after you have finished the input (default: Cursor goes to the field whose ID is configured in *cur\_d*).

**access class**

**One** of the following access classes can be combined with parameter *field\_attr*.

**IO\_ACC\_SIEMENS**

Highest access level (Siemens).

**IO\_ACC\_OEM\_1**

Machine manufacturer.

**IO\_ACC\_OEM\_0**

Service.

**IO\_ACC\_USER**

End user.

**IO\_ACC\_KEY\_SWITCH\_3**

Keyswitch setting 3.

**IO\_ACC\_KEY\_SWITCH\_2**

	Keyswitch setting 2.
	IO_ACC_KEY_SWITCH_1
	Keyswitch setting 1.
	IO_ACC_KEY_SWITCH_0
	Lowest access level keyswitch setting 0.
<i>cur_r, cur_l, cur_d, cur_u,</i>	Identifiers of the fields where the dialog cursor for cursor right, cursor left, cursor down, cursor up changes (KEY_RIGHT, KEY_LEFT, KEY_DOWN, KEY_UP).
<i>acc_class</i>	Number of notepad containing the minimum required access level for write operations. If you enter a notepad no. (value other than 0), the configured access levels (IO_ACC_...) in the <i>field_attr</i> field become inoperative, in which case the access level must be located in the notepad specified here.
<i>refresh</i>	Refresh cycle for visualization (display refresh) and data access (data refresh).  Also refer to Section "Refresh factor".
<i>cursor_txt_id</i>	Text number for an operator info text which is displayed in the dialog line when the dialog cursor enters the field.
<i>v_adr</i>	Data identifier for accessing NC/PLC/MMC data.
<i>v_p1, v_p2, v_p3</i>	Additional parameters for data access.
<i>con</i>	Identifier for conversion of the NC/PLC data, e.g. CON_DECIMAL for decimal representation (see section on data conversion).
<i>con_p1, con_p2, con_p3</i>	Conversion parameters to specify the required data conversion in more detail (refer to the Section Data conversion).

### 3.2.2 Output field - O\_FIELD

<b>Description</b>	The contents of NC variables, PLC variables, MMC variables and MMC methods are made accessible on the operator interface via an output field. The contents are displayed in various modes depending on parameter settings. You cannot position the dialog cursor on fields of this type. No inputs can be made in the output fields.  You can specify the refresh cycle via a parameter.
<b>Syntax</b>	<b>O_FIELD</b> ( <i>id, x, y, w,</i> <i>fc, bc,</i> <i>char_set,</i> <i>field_attr[<i>field_attr...</i>],</i> <i>refresh,</i>

*v\_adr, v\_p1, v\_p2, v\_p3,  
con, con\_p1, con\_p2, con\_p3)*

<b>Parameters</b>	<i>id</i>	Unique identifier for the O_FIELD object in the current module.
	<i>x, y</i>	Position of top left-hand corner of field in pixels relative to the object above it.
	<i>w</i>	Field width in <b>number of characters</b> . This parameter, together with the selected character set and the text attribute, determines the width and height of the field.
	<i>fc</i>	Foreground color (text color). Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
	<i>bc</i>	Background color. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
	<i>char_set</i>	Identifier for character set to be used. CS_SMALL Character set 8 x 12 pixels ( <b>not HPU</b> ) Character set 6 x 8 pixels ( <b>for HPU</b> ) CS_6_9: Character set 6 x 9 pixels (for MMC 100/UOP only); character set must be loaded first using AC/RC_LD_FONT. CS_8_12: (as for CS_SMALL; <b>only for MMC 100/UOP</b> ) CS_19_27: Character set 19 x 27 pixels ( <b>only for MMC 100/UOP</b> ); character set must be loaded first using AC/RC_LD_FONT.
	<i>field_attr</i>	Attribute parameter. Individual attributes can be combined: TEXT_DOUBLE_HEIGHT Characters are displayed in double the height defined in the appropriate selected character set. TEXT_DOUBLE_ZOOMED The characters are doubled in the x and y directions and thus displayed with double line thickness. TEXT_WRITE_LF Linefeed is output as characters "LF". IO_LEFT_ADJUST Text is left-adjusted within the field. IO_CENTRE_ADJUST Text is centered within the field.

If you do not specify either of the two justification attributes (IO\_LEFT\_ADJUST, IO\_CENTRE\_ADJUST), the following standard applies: Text is left-justified in string fields (conversion format CON\_STRING or CON\_STRING\_LIMIT) and left-justified with value formats (all other formats including CON\_ASCII). Only **one** of the two justification attributes may be combined with the attribute word.

<i>refresh</i>	Refresh cycle for visualization (display refresh) and data access (data refresh). See also Section "Refresh factor".
<i>v_adr</i>	Data identifier for accessing NC/PLC/MMC data.
<i>v_p1, v_p2, v_p3</i>	Additional parameters for data access.
<i>con</i>	Identifier for conversion of the NC/PLC data, e.g. CON_DECIMAL for decimal representation (see section on data conversion).
<i>con_p1, con_p2, con_p3</i>	Conversion parameters to specify the required data conversion in more detail (refer to the Section Data conversion).

### 3.2.3 PROGRESS\_BAR

<b>Description</b>	The progress bar represents a value as a bar. If you wish to place a border around the bar, you must configure it as an "empty" rectangle around the bar. The progress bar can be displayed from left to right, from right to left, from bottom to top and from top to bottom.														
<b>Syntax</b>	<b>PROGRESS_BAR</b> ( <i>id, x, y, w, h, fc, bc, sc, attr, ac, rc, min, max, sig100, cl, xp1, xp2, xp3</i> )														
<b>Parameters</b>	<table> <tr> <td><i>id</i></td> <td>Unique identity within the C module in which the progress bar was defined</td> </tr> <tr> <td><i>x, y</i></td> <td>Position of the lefthand upper edge in pixels relative to the element lying above it</td> </tr> <tr> <td><i>w, h</i></td> <td>Width, height in pixels</td> </tr> <tr> <td><i>fc</i></td> <td>Foreground color (color of the bar).</td> </tr> <tr> <td><i>bc</i></td> <td>Background color</td> </tr> <tr> <td><i>sc</i></td> <td>Signal color (color for the part of the bar that exceeds the 100% value)</td> </tr> <tr> <td><i>attr</i></td> <td>Attribute which can assume the following values</td> </tr> </table>	<i>id</i>	Unique identity within the C module in which the progress bar was defined	<i>x, y</i>	Position of the lefthand upper edge in pixels relative to the element lying above it	<i>w, h</i>	Width, height in pixels	<i>fc</i>	Foreground color (color of the bar).	<i>bc</i>	Background color	<i>sc</i>	Signal color (color for the part of the bar that exceeds the 100% value)	<i>attr</i>	Attribute which can assume the following values
<i>id</i>	Unique identity within the C module in which the progress bar was defined														
<i>x, y</i>	Position of the lefthand upper edge in pixels relative to the element lying above it														
<i>w, h</i>	Width, height in pixels														
<i>fc</i>	Foreground color (color of the bar).														
<i>bc</i>	Background color														
<i>sc</i>	Signal color (color for the part of the bar that exceeds the 100% value)														
<i>attr</i>	Attribute which can assume the following values														



**PBAR\_VERTICAL:**

Vertical representation (default: horizontal)

**PBAR\_BACKWARD:**

Display direction reversed (from right to left or from top to bottom) Default is:  
from left to right or from bottom to top

**PBAR\_SIGNAL\_COL:**

Display value within the 100% mark in signal color

<i>ac</i>	Number of a notepad containing the access stage (0->no access stage check)
<i>rc</i>	Refresh cycle: as for IO fields
<i>min</i>	Minimum value for display (value where the bar is invisible)
<i>max</i>	Maximum value for display (value where the bar is colored in completely)
<i>sig100</i>	Signal value (=100% value, after which the display is shown in color)
<i>cl</i>	"Column ID" NC/PLC data or MMC method (as for IO fields)
<i>xp1, xp2, xp3</i>	Extra parameter for data accesses (as with IO fields)

### 3.2.4 TACHOMETER

<b>Description</b>	The tachometer displays a value as in a tachometer. The tachometer can be displayed from left to right or from right to left.														
<b>Syntax</b>	<b>TACHO</b> ( <i>id</i> , <i>x</i> , <i>y</i> , <i>r</i> , <i>fw</i> , <i>fc</i> , <i>bc</i> , <i>sc</i> , <i>attr</i> , <i>ac</i> , <i>rc</i> , <i>min</i> , <i>max</i> , <i>sig100</i> , <i>ci</i> , <i>xp1</i> , <i>xp2</i> , <i>xp3</i> )														
<b>Parameters</b>	<table> <tr> <td><i>id</i></td> <td>Unique identity within the C module in which the progress bar was defined</td> </tr> <tr> <td><i>x, y</i></td> <td>Center point</td> </tr> <tr> <td><i>r</i></td> <td>Radius in pixels</td> </tr> <tr> <td><i>fw</i></td> <td>Border thickness for the rounding-off (FrameWidth)</td> </tr> <tr> <td><i>fc</i></td> <td>Foreground color (color of the pointer and the frame)</td> </tr> <tr> <td><i>bc</i></td> <td>Background color</td> </tr> <tr> <td><i>sc</i></td> <td>Signal color (color of the pointer when the 100% value is exceeded)</td> </tr> </table> <p><i>attr</i> Attribute which can take the following values:</p> <p>TACO_RIGHT_START:</p>	<i>id</i>	Unique identity within the C module in which the progress bar was defined	<i>x, y</i>	Center point	<i>r</i>	Radius in pixels	<i>fw</i>	Border thickness for the rounding-off (FrameWidth)	<i>fc</i>	Foreground color (color of the pointer and the frame)	<i>bc</i>	Background color	<i>sc</i>	Signal color (color of the pointer when the 100% value is exceeded)
<i>id</i>	Unique identity within the C module in which the progress bar was defined														
<i>x, y</i>	Center point														
<i>r</i>	Radius in pixels														
<i>fw</i>	Border thickness for the rounding-off (FrameWidth)														
<i>fc</i>	Foreground color (color of the pointer and the frame)														
<i>bc</i>	Background color														
<i>sc</i>	Signal color (color of the pointer when the 100% value is exceeded)														

	Display starts at bottom right and goes in counter-clockwise direction (default: bottom left, clockwise)
	TACO_CENTRE_START:
	Display starts in the top center and goes in positive and negative directions
	TACHO_SIGNAL_COL:
	Display value within the 100% mark in signal color
<i>ac</i>	Number of a notepad containing the access stage (0->no access stage check)
<i>rc</i>	Refresh cycle: as for IO fields
<i>min</i>	Minimum value
<i>max</i>	Maximum value for display
<i>sig100</i>	Signal value (=100% value, after which the pointer is shown in signal color)
<i>ci</i>	"Column ID" NC/PLC data or MMC methods (as for IO fields)
<i>xp1, xp2, xp3</i>	Supplementary parameters for data accesses (as for IO fields)

### 3.2.5 Input field - I\_FIELD

<b>Description</b>	<p>The contents of NC variables, PLC variables, MMC variables and MMC methods can be modified on the operator interface via an input field.</p> <p>This field type basically behaves like an IO field without the refresh operation. You can specify the sequence in which the fields are processed by a parameter setting.</p>
<b>Syntax</b>	<p><b>I_FIELD</b> (<i>id</i>, <i>x</i>, <i>y</i>, <i>w</i>,  <i>fc</i>, <i>bc</i>,  <i>char_set</i>,  <i>field_attr</i>[<i>field_attr...</i>] [<i>access class</i>],  <i>cur_r</i>, <i>cur_l</i>, <i>cur_d</i>, <i>cur_u</i>,  <i>acc_class</i>,  <i>cursor_txt_id</i>,  <i>v_adr</i>, <i>v_p1</i>, <i>v_p2</i>, <i>v_p3</i>,  <i>con</i>, <i>con_p1</i>, <i>con_p2</i>, <i>con_p3</i>)</p>
<b>Parameters</b>	<p><i>id</i> Unique identifier for the I_FIELD object in the current module.</p> <p><i>x, y</i> Position of top left-hand corner of field in pixels relative to the object above it.</p> <p><i>w</i> Field width in <b>number of characters</b>. This parameter, together with the selected character set and the text attribute, determines the width and height of the field.</p>

<i>fc</i>	Foreground color (text color). Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
<i>bc</i>	Background color. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
<i>char_set</i>	Identifier for character set to be used. CS_SMALL Character set 8 x 12 pixels ( <b>not HPU</b> ) Character set 6 x 8 pixels ( <b>for HPU</b> ) CS_6_9: Character set 6 x 9 pixels (for MMC 100/UOP only); character set must be loaded first using AC/RC_LD_FONT. CS_8_12: (as for CS_SMALL; <b>only for MMC 100/UOP</b> ) CS_19_27: Character set 19 x 27 pixels ( <b>only for MMC 100/UOP</b> ); character set must be loaded first using AC/RC_LD_FONT.
<i>field_attr</i>	Attribute parameter. Individual attributes can be combined: TEXT_DOUBLE_HEIGHT The field height is twice the height defined in the appropriate selected character set. TEXT_DOUBLE_ZOOMED The field height and width are doubled. IO_INPUT_DISABLE You can select the field with the dialog cursor, but you cannot open it for editing. IO_CURSOR_DISABLE Field is skipped by the dialog cursor. DIA_CURSOR_STOP The cursor remains on this field after you have finished the input (default: Cursor goes to the field whose ID is configured in <i>cur_d</i> .
<i>access class</i>	<b>One</b> of the following access classes can be combined with <i>field_attr</i> . IO_ACC_SIEMENS Highest access level (Siemens). IO_ACC_OEM_1 Machine manufacturer. IO_ACC_OEM_0 Service.

	IO_ACC_USER	End user.
	IO_ACC_KEY_SWITCH_3	Keyswitch setting 3.
	IO_ACC_KEY_SWITCH_2	Keyswitch setting 2.
	IO_ACC_KEY_SWITCH_1	Key switch setting 1 (value 6).
	IO_ACC_KEY_SWITCH_0	Lowest access level keyswitch setting 0 (value 7).
<i>cur_r, cur_l, cur_d, cur_u,</i>		Identifiers of the fields where the dialog cursor for cursor right, cursor left, cursor down, cursor up changes (KEY_RIGHT, KEY_LEFT, KEY_DOWN, KEY_UP).
<i>acc_class</i>		Number of notepad containing the minimum required access level for write operations. If you enter a notepad no. (value other than 0), the configured access levels (IO_ACC_...) in the <i>field_attr</i> field become inoperative, in which case the access level must be located in the notepad specified here.
<i>cursor_txt_id</i>		Text number for an operator info text which is displayed in the dialog line when the dialog cursor enters the field.
<i>v_adr</i>		Data identifier for accessing NC/PLC/MMC data.
<i>v_p1, v_p2, v_p3</i>		Additional parameters for data access.
<i>con</i>		Identifier for conversion of the NC/PLC data, e.g. CON_DECIMAL for decimal representation (see section on data conversion).
<i>con_p1, con_p2, con_p3</i>		Conversion parameters to specify the required data conversion in more detail (refer to the Section Data conversion).

### 3.2.6 Graphic list field for cursor - CUR\_PICT\_FIELD

<b>Description</b>	A static object list is output once or cyclically in a graphic list field, controlled by the value of a linked variable from the NCK/PLC or MMC. Unlike the PICT_FIELD element, you can position the cursor on this field (MMC 100/UOP only)..
<b>Syntax</b>	<b>CUR_PICT_FIELD</b> ( <i>id, x, y, w, h, cur_r, cur_l, cur_d, cur_u, bc, attr, refresh, cursor_txt, id, v_ade, v_p1, v_p2, v_p3</i> )
<b>Parameters</b>	<p><i>id</i> Unique identifier for the CUR_PICT_FIELD object in the current module.</p> <p><i>x, y</i> Basic position of graphic. Top left-hand corner of field in pixels relative to the higher-level object.</p> <p><i>w, h</i> Width and height of the field in pixels. When refreshed, this area is always cleared in the background color of the field.</p> <p><i>cur_r, cur_l, cur_d, cur_u</i> Identifiers of fields to which the dialog cursor goes on cursor-right, cursor-left, cursor-down, cursor-up (KEY_RIGHT, KEY_LEFT, KEY_DOWN, KEY_UP).</p> <p><i>bc</i> Background color. Colors and gray scales are system-dependent (see section on Colors for valid values).</p> <p><i>attr</i> Attribute: The only valid settings are 0 and IO_CURSOR_DISABLE. This setting prevents the cursor from going to the field.</p> <p><i>refresh</i> Refresh cycle for visualization (display refresh) and data access (data refresh). Also refer to Section "Refresh factor".</p> <p><i>cursor_txt_id</i> Text number for an operator info text which is displayed in the dialog line when the dialog cursor enters the field.</p> <p><i>v_adr</i> Data identifier for accessing data of the NC/PLC/MMC (see also Section "Data identifier"):  <b>This data access operation must return the identifier for the static object list (obl_id) to be displayed.</b></p> <p><i>v_p1, v_p2, v_p3</i> Supplementary parameters for data access.</p>

### 3.2.7 Single and multiple option boxes - CHECK\_FIELD

**Description** You can display single-option  $\bullet/\odot$  and multiple-option boxes (checkboxes)  $\boxtimes/\square$  with this element. These enable you to simulate control elements as they are used, for example, in MS Windows. The fields are selected or deselected using the Select key..

The field displays the status of a single bit or a bit combination of an item of data on the NC/PLC/MMC. When you click the Select key (with the cursor positioned on the field), the state of the bit(s) is inverted. The fields are then displayed as active  $\boxtimes$ , "Y", "1",  $\boxtimes$  or  $\odot$ ) if the specified NCK/PLC/MMC data (*v\_adr*) produces **1** when ANDed with the specified bit mask (*bit\_mask*); or, if the attribute CHECK\_ZERO\_ACTIV is set and the AND operation returns a **0**.

**Syntax** **CHECK\_FIELD** (*id*, *x*, *y*, *w*,  
*fc*, *bc*,  
*char\_set*,  
*field\_attr*[*field\_attr...*] [*access class*],  
*cur\_r*, *cur\_l*, *cur\_d*, *cur\_u*,  
*acc\_class*,  
*refresh*,  
*bit\_mask*,  
*cursor\_txt\_id*,  
*v\_adr*, *v\_p1*, *v\_p2*, *v\_p3*)

**Parameters**

<i>id</i>	Unique identifier for the CHECK_FIELD object in the current module.
<i>x</i> , <i>y</i>	Position of top left-hand corner of field in pixels relative to the higher-level object.
<i>w</i>	<b>Only for CHECK_CROSS_BLANK:</b> Field width in <b>number of characters</b> . The width and height of the field (1 would normally be a meaningful setting) are determined by this parameter and the selected character set and text attribute.
<i>fc</i>	Foreground color (text color). Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
<i>bc</i>	Background color. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
<i>char_set</i>	Identifier for character set to be used. CS_SMALL Character set 8 x 12 pixels ( <b>not HT6</b> ) Character set 6 x 8 pixels ( <b>for HPU</b> ) CS_6_9: Character set 6 x 9 pixels ( <b>only for MMC 100/UOP</b> ); character set must be loaded first using AC/RC_LD_FONT. CS_8_12: (as for CS_SMALL; <b>only for MMC 100/UOP</b> )

## CS\_19\_27:

Character set 19 x 27 pixels (**only for MMC 100/UOP**); character set must be loaded first using AC/RC\_LD\_FONT.

*field\_attr*

Attribute parameter. The individual attributes can be combined by OR'ing:

## IO\_INPUT\_DISABLE

You can select the field with the dialog cursor, but you cannot open it for inputs.

## IO\_CURSOR\_DISABLE

Field is skipped by the dialog cursor.

## DIA\_CURSOR\_STOP

The cursor remains on this field after you have finished the input (default: Cursor goes to the field whose ID is configured in *cur\_d*).

## CHECK\_ZERO\_ACTIVE

The value "0" is used as an active value and results in display of ☑, "Y", "1", ☒ or ⊙.

**The following attributes determine the field layout and can only be used exclusively.**

## CHECK\_TICK\_BLANK

Displays the value as a tick (☑) or blank field.

## CHECK\_YES\_NO

Displays the value as Yes "Y" or No "N" - as a function of language.

## CHECK\_ONE\_ZERO

Displays the value as a "1" or "0".

## CHECK\_CROSS\_BLANK

Displays the value as a cross (☒) or blank field (□).

## CHECK\_BUTTON\_BLANK

Displays the value as a button (⊙) or blank circle (●).

*access class*

**One** of the following access classes can be combined with *field\_attr*.

## IO\_ACC\_SIEMENS

Highest access level (Siemens).

## IO\_ACC\_OEM\_1

Machine manufacturer.

## 3.2 Dynamic display elements – dialog fields

	IO_ACC_OEM_0	Service.
	IO_ACC_USER	End user.
	IO_ACC_KEY_SWITCH_3	Keyswitch setting 3.
	IO_ACC_KEY_SWITCH_2	Keyswitch setting 2.
	IO_ACC_KEY_SWITCH_1	Keyswitch setting 1.
	IO_ACC_KEY_SWITCH_0	Lowest access level keyswitch setting 0.
<i>cur_r, cur_l, cur_d, cur_u,</i>		Identifiers of the fields where the dialog cursor for cursor right, cursor left or cursor down, cursor up changes (KEY_RIGHT, KEY_LEFT, KEY_DOWN, KEY_UP).
<i>acc_class</i>		Number of notepad containing the minimum required access level for write operations. If you enter a notepad no. (value other than 0), configured access levels (IO_ACC_...) in the <i>item_attr</i> parameter become inoperative, in which case the access level must be located in the notepad specified here. Valid values for the access level are defined in parameter <i>item_attr</i> .
<i>refresh</i>		Refresh cycle for visualization (display refresh) and data access (data refresh). Also refer to Section Refresh factor.
<i>bit_mask</i>		Bit mask which is ANDed with the value supplied under <i>v_adr</i> . The length of the bit mask is one byte or one UWORD.
<i>cursor_txt_id</i>		Text number for an operator info text which is displayed in the dialog line when the dialog cursor enters the field.
<i>v_adr</i>		Data identifier for accessing NC/PLC/MMC data.
<i>v_p1, v_p2, v_p3</i>		Additional parameters for data access.



### 3.2.8 Edit field /NC – data overview – EDIT\_FIELD

**Description**

The purpose of the edit field is to display  
 an **NC file**,  
 an **NC directory**,  
 a **DOS file** (not for OP 030)..

The edit field is also used to display the current block, the compensation block and internal lists.

The edit field attributes determine the display mode.

**Syntax**

**EDIT\_FIELD** (*id*, *x*, *y*, *w*, *h*,  
*fc*, *bc*, *b/c*  
*char\_set*,  
*edit\_attr*[*edit\_attr...*],  
*max\_bl\_len*,  
*line\_dist*,  
*name\_nb*)

**Parameters**

*id* Unique identifier for the EDIT\_FIELD object in the current module.




---

#### Important

If you use a scrollbar for the editor, you must pay heed to the relationship between the scrollbar ID and the editor ID, i.e. the scrollbar number must be identical to the editor ID modulo 8.

(e.g. Editor-Id=16 => Scrollbar number=0  
 Editor-Id=19 => Scrollbar number=3)

---

<i>x</i> , <i>y</i>	Position of top left-hand corner of field in pixels relative to the object above it.
<i>w</i>	Number of columns.
<i>h</i>	Number of rows.
<i>fc</i>	Foreground color (text color). Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
<i>bc</i>	Background color. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
<i>b/c</i>	Color for the current block (in 'current block display' mode only). Colors and gray scales are system-dependent (see Section <b>Colors</b> for valid values). <i>char_set</i> Identifier for character set to be used.
	CS_SMALL
	Character set 8x 12 pixels ( <b>not HT6</b> )
	Character set 6 x 8 pixels ( <b>for HPU</b> )

CS\_6\_9:

Character set 6 x 9 pixels (for MMC 100/UOP only); character set must be loaded first using AC/RC\_LD\_FONT.

CS\_8\_12:

(as for CS\_SMALL; **only for MMC 100/UOP**)

CS\_19\_27:

Character set 19 x 27 pixels (**only for MMC 100/UOP**); character set must be loaded first using AC/RC\_LD\_FONT.

*edit\_attr*

Edit field attributes.

An edit field attribute comprises an attribute component for the display mode and a parameter for the editor.

#### **1. Display mode:**

To display a file or directory, you must set the edit field attributes and activate an appropriate action for the actual display. The following different modes are used:

**Editor mode** for displaying NC files (default).

(Call: refer to {AC|RC}\_PP\_EDIT\_OPEN)

The identifier for this default setting is '0'. An editor parameter must be specified in order to generate the complete attribute. This is the reason that information is not required here.

**Directory mode** for displaying NC directories.

(call, refer to {AC|RC}\_PP\_EDIT\_OPEN and {AC|RC}\_NC\_DIR)

The identifier for this mode is E\_NC\_DIR.

**DOS file editor mode** for displaying DOS files.

(Call: refer to {AC|RC}\_PP\_EDIT\_OPEN)

Display mode parameters can be used exclusively only.

The identifier for this mode is E\_DOSFILE.

#### **2nd Editor parameter:**

E\_READ\_ONLY

The editor is in display mode only, it cannot be written.

E\_CURSOR\_OFF

No cursor display.

E\_FREE\_DEFINE

Free definition of columns.

E\_LINE\_INVERS

The cursor is displayed as a selection bar, i.e. the whole line is inverted; useful for directory displays.

**E\_NOT\_DISPLAY** (for HPU only)

The content of the EDIT\_FIELD is not displayed on the screen.  
Useful for processing an object (e.g. part program) in the background

**E\_SCROLLBAR**

The scrollbar is automatically updated by the editor if you make the appropriate reference entry when configuring the EDIT\_FIELD.

**E\_SHOW\_NO\_ACC**

Suppresses display of the 'shared' ID in directory overviews.

**E\_ACTUAL\_DATA****Note**

The notepad entry for directories (*name\_nb+1*) must always include a pointer to a string!

This is not the case with files! A number is written to the notepad (*name\_nb + 1*).

**i.e. the notepad (*name\_nb*) must be preset to AC|RC\_TXT\_NB\_SINGLE (only for MMC 100/UOP).**

The editor parameters can be combined with one another and with **one** display mode parameter.

Meaningful combinations:

Desired functions	Display mode	Editor parameters
Display NC file, – editable	---	E_SCROLLBAR   E_ACTUAL_DATA
Display NC directory	E_NC_DIR	E_SCROLLBAR   E_LINE_INVERS   E_ACTUAL_DATA
Display DOSfile	E_DOSFILE	E_SCROLLBAR   E_LINE_INVERS

MMC100/UOP, SW 5 and later:

Desired functions	Display mode	Editor parameters
Display NC file,– editable	---	E_SCROLLBAR   E_ACTUAL_DATA
Display NC directory	E_NC_DIR	E_SCROLLBAR   E_LINE_INVERS   E_ACTUAL_DATA
Display DOS file, editable	---	E_ACTUAL_DATA   E_SCROLLBAR   E_LINE_INVERS

## 3.2 Dynamic display elements – dialog fields

<i>max_bl_len</i>	Maximum length of a block (block up to and including LF) accepted by the editor (including copying and insertion). The highest possible value is stored in ED_MAX_RECORD_LEN. This should be passed as the default here. A value of zero causes malfunctions.
<i>line_dist</i>	Distance between lines in pixels.
<i>name_nb</i>	Communication notepad between configuration and editor. The name of the file to be displayed must be saved in this notepad (call, refer to AC RC_SET_TXT_NB(_SINGLE). This notepad is used as a transfer value or return value of the editor depending on the display mode.

**Note**

The notepad entry after the transferred entry (*name\_nb*) is also used by the editor as a return parameter when E\_ACTUAL\_DATA is parameterized. For this reason, one notepad should always be reserved after *name\_nb*.

*name\_nb* and *name\_nb+1* (when E\_ACTUAL\_DATA is used and directories are displayed) must always contain a pointer to a string. The editor always writes the result to this pointer. The pointer must be assigned using a C function or using the (re-)action routine {AC|RC}\_SET\_TXT\_NB.

This is not necessary when files are displayed. It need only be reserved.

Display mode	<i>name_nb</i>	<i>name_nb+1</i> for E_ACTUAL_DATA
Edit mode and DOS edit mode	to editor: (NC) file name	from editor: String with block number
Directory mode	from editor: String with selected file (marked by the cursor position in the display)	from editor: String with access rights (bit field)

**Note**

You can only configure one edit field in a window. If you configure several windows of an application that can be displayed simultaneously or overlapping with edit fields, then you must give the edit fields different identifications (IDs).

**3.2.9 Edit\_feld\_32**

New attributes have been inserted for the edit field. These attributes can only be specified when configuring the EDIT\_FIELD\_32.

**E\_OUTPUT\_INVISIBLE**

Data is not displayed.

From HMI-Embedded 06.02 onwards.

**E\_READ\_ONLY\_LINESIZE**

Only the number of configured lines is read.  
From HMI-Embedded 06.02 onwards.

#### **E\_SHOW\_LF\_AS\_BLANK**

The end of line is not represented as line feed.  
From HMI-Embedded 06.02 onwards.

#### **E\_NC\_VERSION (internal)**

An NC directory is displayed including the version string.  
From HMI-Embedded 06.04.04 onwards.

#### **E\_NC\_SOURCE (internal)**

An NC directory is displayed with the sources.

#### **E\_NC\_PACKAGE (internal)**

An NC directory is displayed as package overview.  
From HMI-Embedded 06.04.04 onwards.

#### **E\_NOT\_DISPLAY\_EOF**

The display of the string "=== EOF ===" at the end of the file should be suppressed.  
From HMI-Embedded 06.04.04 onwards.

#### **E\_NO\_CURSOR\_INACTIV**

If the window with the edit field is not active, then normally, the actual line is shown in gray. The operator still always sees the line where the cursor is presently located. The attribute should be set if this is to be prevented.  
From HMI-Embedded 06.04.04 onwards.

#### **E\_NO\_MESSAGES**

In its functions, the Editor outputs various messages – for example "Please wait, data are being saved!". This attribute can be used to suppress the display of messages.  
From HMI-Embedded 06.04.04 onwards.

#### **E\_NO\_CALC\_CHAR\_WIDTH**

As a result of the various resolutions, it is possible that more characters fit in a line than were originally configured. However, if precisely the number of configured characters per line are to be output, independent of the resolution, then this attribute must be set.  
From HMI-Embedded 06.04.04 onwards.

#### **E\_SIZE\_IS\_PIXEL**

The size of the edit field can now also be specified in pixels. Previously, the data was entered in the number of characters per line (width) and the number of lines per page (height).

Now, these values can also be specified in pixels. The appropriate, optimum number of characters per line and lines per page are calculated by the editor. If this bit is set, then bit E\_NO\_CALC\_CHAR\_WIDTH is ignored.  
From HMI-Embedded 06.04.06 onwards.

### 3.2.10 Table - TABLE

<b>Description</b>	<p>A table is a composed dynamic display element. The contents of multiple NC variables, PLC variables or MMC variables can be displayed in the table either in columns or rows. A table consists of</p> <ul style="list-style-type: none"> <li>• The table definition (layout definition, reference to the column definition and the applicable data server link TABLE),</li> <li>• The table column definition (TAB_COLUMN) and</li> <li>• The table data element definition (TAB_ITEM).</li> </ul>																						
<b>Syntax</b>	<p><b>TABLE</b> (<i>table_id</i>, <i>x</i>, <i>y</i>, <i>w</i>, <i>h</i>,  <i>row_dist</i>, <i>row_cnt</i>,  <i>bc</i>,  <i>sb_nr</i>,  <i>dummy_nb_nr</i>,  <b>OBJECT_LIST_PTR</b> (<i>table_col_list_id</i>),  <b>OBJECT_LIST_PTR</b> (<i>graphic_list_id</i>),</p> <p><i>ovl_id</i>,  <i>serv_fkt_id</i>,  <i>serv_fkt_p1</i>, <i>serv_fkt_p2</i>, <i>serv_fkt_p3</i>, <i>serv_fkt_p4</i>)</p>																						
<b>Parameters</b>	<table border="0"> <tr> <td style="vertical-align: top;"><i>table_id</i></td> <td>Unique identifier of the table in this module.</td> </tr> <tr> <td style="vertical-align: top;"><i>x</i>, <i>y</i></td> <td>Position of top left-hand corner of field in pixels relative to the higher-level element.</td> </tr> <tr> <td style="vertical-align: top;"><i>w</i>, <i>h</i></td> <td>Table width and table height in pixels.</td> </tr> <tr> <td style="vertical-align: top;"><i>row_dist</i></td> <td>Distance between lines in pixels. (recommended for OP030: STD_ROW_DISTANCE).</td> </tr> <tr> <td style="vertical-align: top;"><i>row_cnt</i></td> <td>max. number of lines in the window. (recommended for OP030: STD_ROW_MAX).</td> </tr> <tr> <td style="vertical-align: top;"><i>bc</i></td> <td>Background color of table. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).</td> </tr> <tr> <td style="vertical-align: top;"><i>sb_nr</i></td> <td>Scrollbar number of scrollbar that is to be activated by the table block. Assign a scrollbar by specifying the relevant scrollbar number.</td> </tr> <tr> <td style="vertical-align: top;"><i>dummy_nb_nr</i></td> <td>MMC 100/UOP only, must always be 0.</td> </tr> <tr> <td style="vertical-align: top;"><i>table_col_list_id</i></td> <td>Unique identifier for an object list with associated column entries for the table. <b>Only table column objects (TAB_COLUMN) are allowed to be defined in this object list.</b></td> </tr> <tr> <td style="vertical-align: top;"><i>graphic_list_id</i></td> <td>Unique identifier of an object list which, for this table, additional static display elements are contained. The object list is output before the table.</td> </tr> <tr> <td style="vertical-align: top;"><i>ovl_id</i></td> <td>For OP 030, always = 0 !</td> </tr> </table>	<i>table_id</i>	Unique identifier of the table in this module.	<i>x</i> , <i>y</i>	Position of top left-hand corner of field in pixels relative to the higher-level element.	<i>w</i> , <i>h</i>	Table width and table height in pixels.	<i>row_dist</i>	Distance between lines in pixels. (recommended for OP030: STD_ROW_DISTANCE).	<i>row_cnt</i>	max. number of lines in the window. (recommended for OP030: STD_ROW_MAX).	<i>bc</i>	Background color of table. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).	<i>sb_nr</i>	Scrollbar number of scrollbar that is to be activated by the table block. Assign a scrollbar by specifying the relevant scrollbar number.	<i>dummy_nb_nr</i>	MMC 100/UOP only, must always be 0.	<i>table_col_list_id</i>	Unique identifier for an object list with associated column entries for the table. <b>Only table column objects (TAB_COLUMN) are allowed to be defined in this object list.</b>	<i>graphic_list_id</i>	Unique identifier of an object list which, for this table, additional static display elements are contained. The object list is output before the table.	<i>ovl_id</i>	For OP 030, always = 0 !
<i>table_id</i>	Unique identifier of the table in this module.																						
<i>x</i> , <i>y</i>	Position of top left-hand corner of field in pixels relative to the higher-level element.																						
<i>w</i> , <i>h</i>	Table width and table height in pixels.																						
<i>row_dist</i>	Distance between lines in pixels. (recommended for OP030: STD_ROW_DISTANCE).																						
<i>row_cnt</i>	max. number of lines in the window. (recommended for OP030: STD_ROW_MAX).																						
<i>bc</i>	Background color of table. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).																						
<i>sb_nr</i>	Scrollbar number of scrollbar that is to be activated by the table block. Assign a scrollbar by specifying the relevant scrollbar number.																						
<i>dummy_nb_nr</i>	MMC 100/UOP only, must always be 0.																						
<i>table_col_list_id</i>	Unique identifier for an object list with associated column entries for the table. <b>Only table column objects (TAB_COLUMN) are allowed to be defined in this object list.</b>																						
<i>graphic_list_id</i>	Unique identifier of an object list which, for this table, additional static display elements are contained. The object list is output before the table.																						
<i>ovl_id</i>	For OP 030, always = 0 !																						

<i>serv_fkt_id</i>	Identifier for data server function. (for OP 030: only VAR_SERVER possible)
<i>serv_fkt_p1 - 4</i>	Parameter for data server. (for OP 030 and VAR_SERVER, all 0 !)

**Note**                      **Only one table can be defined in this object list!**

### 3.2.11 Table column - TAB\_COLUMN

**Description**                      TAB\_COLUMN specifies exactly one column of a table. A max. of 6 (OP030) or 7 (MMC 100/EBF) columns are possible for each table. These six/seven TAB\_COLUMN entries must be contained in a shared object list.

(BEGIN\_OBJECT\_LIST(*table\_col\_list\_id*)

....

END\_OBJECT\_LIST(*table\_col\_list\_id*))

**Only table columns (TAB\_COLUMN) are allowed to be defined in this object list.**

**Syntax**                              **TAB\_COLUMN** (*table\_col\_id*, *x*, *y*, *w*,  
*char\_set*,  
*fc*, *bc*,  
*table\_col\_attr*[*table\_col\_attr*...],  
*field\_typ*,  
*refresh*,  
*column\_index*,  
*column\_attr*,  
**OBJECT\_LIST\_PTR** (*tab\_item\_list\_id*))

**Parameters**

<i>table_col_id</i>	Unique identifier of table column in this module.
<i>x</i> , <i>y</i>	Position of top left-hand corner of field in pixels relative to the table position.
<i>w</i>	Column width in number of characters.
<i>char_set</i>	Character set:  CS_SMALL  Small character set (8 x 12 pixels)
<i>fc</i>	Foreground color (text color). Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
<i>bc</i>	Background color. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
<i>field_typ</i>	Column type – output-input, input/output field or selection fields

**OP\_OUTPUT\_FIELD**

Output field only. No cursor display and no input possible, the values are only output.

**OP\_INPUT\_FIELD**

Only input field – with cursor – input possible, values supplied from MMC, NCK or PLC are not displayed.

**OP\_IO\_FIELD**

Input and output field with cursor. Input possible, values from MMC, NCK or PLC are also displayed.

**OP\_CHECK\_FIELD**

Single / and multiple selection fields / can be displayed using this option. These enable you to simulate control elements as they are used, for example, in MS Windows. For more information see CHECK\_FIELD.

*table\_col\_attr*

Attribute parameter. The individual attributes can be ORed: for attributes or possible OR operations, please refer to the description of corresponding fields. E.g. for an output field (OP\_OUTPUT\_FIELD) see the attribute description of the output field (OUTPUT\_FIELD); for a selection field (OP\_CHECK\_FIELD) see the attribute description for the selection field (CHECK\_FIELD), etc.).

The following attributes are permissible:

**OP030:**

TEXT\_DOUBLE\_HEIGHT  
 TEXT\_DOUBLE\_ZOOMED  
 IO\_CALCULATOR\_OFF  
 IO\_INPUT\_DISABLE  
 IO\_OUTPUT\_DISABLE  
 IO\_CURSOR\_DISABLE  
 IO\_UNCOND\_REFRESH  
 IO\_FRAME\_CURSOR  
 IO\_INV\_FRAME\_CURSOR  
 DIA\_CURSOR\_STOP  
 IO\_RIGHT\_ADJUST  
 IO\_CENTRE\_ADJUST  
 IO\_LEFT\_ADJUST

**MMC100/EBF:**

TEXT\_DOUBLE\_HEIGHT  
 TEXT\_DOUBLE\_ZOOMED  
 IO\_CALCULATOR\_OFF  
 IO\_INPUT\_DISABLE  
 DIA\_CURSOR\_STOP  
 TEXT\_RIGHT\_ADJUST  
 TEXT\_CENTRE\_ADJUST  
 TEXT\_PIXEL\_ADJUST  
 IO\_CENTRE\_ADJUST  
 IO\_LEFT\_ADJUST  
 IO\_EDITOR\_SCROLL



*refresh* Refresh cycle for visualization (display refresh) and data access (data refresh).




---

#### Important

If there is a TAB\_ITEM for this column containing data access to drive data, you must not enter a refresh factor here! For example, all data of the block service values drives (P\_V\_S...).

---

*column\_index* Index for the table columns. Each index may only be used once within a table, otherwise contents are overwritten. This index is also an identifier for the data server (according to parameter settings). There are pre-defined identifiers for the logical table indexes (COL\_1 ... COL\_6, COL7).

*column\_attr* Attribute for the table columns.

For VAR\_SERVER (refer to *serv\_fkt\_id* in TABLE) the following applies:

#### VAR\_UNIT:

Column in which the physical units are displayed (if assigned in TAB\_ITEM).

#### VAR\_TEXT:

Column with text (if assigned in TAB\_ITEM).  
If this identifier is set, then only one OBJECT\_LIST (*tab\_item\_list\_id*) is used for two (or if both identifiers are used), three columns.  
In this TAB\_ITEM, then a text number can be specified in the *txt\_id\_1* that is displayed in this column.

If the attribute VAR\_TEXT or VAR\_UNIT is set in this table column, then only the appropriate text (no variable values) are displayed in this table column.

A dedicated column must be defined for the variable value.

*table\_item\_list\_id* Unique identifier for an object list with associated table elements for precisely this column. All entries in this object list are displayed in this table column. These are essentially the line entries for this column.  
**Only table element objects (TAB\_ITEM) may be defined in this object list.**

### 3.2.12 Table data element – line entry - TAB\_ITEM

<b>Description</b>	<p><b>This configuration list is only used for the table data server VAR_SERVER.</b></p> <p>Exactly one table data element for output in a column is defined via TAB_ITEM. All table data elements for a column must be contained in a shared object list.</p> <pre>BEGIN_OBJECT_LIST(<i>tab_item_list_id</i>) .... END_OBJECT_LIST(<i>tab_item_list_id</i>)</pre> <p><b>Only table data elements (TAB_ITEMS) are allowed to be defined in this object list.</b></p> <p>With specific parameter settings (in <i>v_item_index</i>), a table data element can result in more than one line outputs in one column, (e.g. actual value output for all geometry axes via a table item entry).</p> <p>Cross references for fixed text output (<i>txt_id_l</i>) and for the physical unit (<i>phys_unit</i>) can result in output in table columns identified via VAR_TEXT / VAR_UNIT.</p>
<b>Syntax</b>	<pre><b>TAB_ITEM</b> (<i>tab_item_id</i>,             <i>v_adr</i>, <i>v_p1</i>, <i>v_p2</i>, <i>v_p3</i>,             <i>con</i>, <i>con_p1</i>, <i>con_p2</i>, <i>con_p3</i>,             <i>lim_check</i>, <i>lim_l</i>, <i>lim_h</i>,             <i>item_attr</i>[<i>item_attr...</i>],             <i>acc_class</i>,             <i>txt_id_l</i>,             <i>phys_unit</i>,             <i>cursor_txt_id</i>,             <i>item_index_typ</i>,             <i>v_item_index</i>,             <i>v_item_index_p1</i>, <i>v_item_index_p2</i>, <i>v_item_index_p3</i>)</pre>
<b>Parameters</b>	<p><i>tab_item_id</i>      Unique identifier for the TAB_ITEM object in the current module.</p> <p><i>v_adr</i>              Data identifier for accessing NC/PLC/MMC data (variables). This address remains the basic address for the data that is displayed in the column via this table item.</p> <p><i>v_p1</i>, <i>v_p2</i>, <i>v_p3</i>      Additional parameters for data access.</p> <p><i>con</i>                Identifier for conversion of the NC/PLC data (see section on data conversion).</p> <p><i>con_p1</i>, <i>con_p2</i>, <i>con_p3</i>      Conversion parameters to specify the required data conversion in more detail (refer to the Section Data conversion).</p>

<i>lim_check</i>	<p>Type of limit value check for input. When the value range is violated, the limit values are displayed in the dialog line. Possible checks:</p> <p>CHECK_OFF No limit value check.</p> <p>CHECK_ON Check for violation of the under and upper limits.</p> <p>CHECK_PLUS Check, whether positive.</p> <p>CHECK_MIN Check only against a lower limit value.</p> <p>CHECK_MAX Check only against an upper limit value.</p>
<i>lim_l</i>	Lower limit value. The limit value type must correspond to data type DOUBLE (e.g. 45.4).
<i>lim_h</i>	Upper limit value. The limit value type must also correspond to data type DOUBLE (e.g. 1234.56).
<i>item_attr</i>	<p>TAB_TXT_INDEX attaches an absolute table numbering starting with 0 at the column display.</p> <p>TAB_TXT_INDEX_1 attaches an absolute table numbering starting with 1 at the column display. (you can use only TAB_TXT_INDEX or TAB_TXT_INDEX_1 they are mutually exclusive)</p> <p>TAB_TXT_BRACKETS places the table number in square brackets [nr] . Only in conjunction (i.e. OR'ing) with TAB_TXT_INDEX or TAB_TXT_INDEX_1.</p>
<i>Access classes</i>	<p><b>One</b> of the following access classes can be OR'ed with <i>item_attr</i>.</p> <p>IO_ACC_SIEMENS Highest access level (Siemens).</p> <p>IO_ACC_OEM_1 Machine manufacturer.</p> <p>IO_ACC_OEM_0 Service.</p> <p>IO_ACC_USER End user.</p> <p>IO_ACC_KEY_SWITCH_3 Keyswitch setting 3.</p> <p>IO_ACC_KEY_SWITCH_2 Keyswitch setting 2.</p> <p>IO_ACC_KEY_SWITCH_1 Keyswitch setting 1.</p>

	IO_ACC_KEY_SWITCH_0
	Lowest access level keyswitch setting 0.
<i>acc_class</i>	Number of notepad containing the minimum required access level for write operations. If you enter a notepad no. (value other than 0), configured access levels (IO_ACC_...) in the item_attr parameter become inoperative, in which case the access level must be located in the notepad specified here. Valid values for the access level are defined in parameter item_attr.
<i>txt_id_l</i>	If you specify a text number here, this text is displayed in the column with the VAR_TEXT identifier (see also section on table columns TAB_COLUMN). If you do not want any text output, specify 0 here.
<i>phys_unit</i>	<p>If you enter a symbolic definition for the physical unit, then depending on the data to be displayed (<i>v_adr</i>) the real unit is displayed in the column with identifier VAR_UNIT (if a column is configured).</p> <p>Possible symbolic values for the physical unit are:</p> <p>NO_UNIT No unit</p> <p>UNIT_POSN_LIN Linear position (mm or inch)</p> <p>UNIT_POSN_ROT Rotary position (degrees)</p> <p>UNIT_VELO_LIN Linear velocity (mm/min or inch/min)</p> <p>UNIT_VELO_ROT Rotary velocity (rpm)</p> <p>UNIT_ACCEL_LIN Linear acceleration (m/s<sup>2</sup> or inch/s<sup>2</sup>)</p> <p>UNIT_ACCEL_ROT Rotary acceleration (rev/s<sup>2</sup>)</p> <p>UNIT_JERK Linear or rotary jerk (m/s<sup>3</sup>, inch/s<sup>3</sup> or rev/s<sup>3</sup>); dependent on axis type (with axis-specific values).</p> <p>UNIT_JERK_LIN Linear jerk (m/s<sup>3</sup> or inch/s<sup>3</sup>)</p> <p>UNIT_JERK_ROT Rotary jerk (rev/s<sup>3</sup>)</p> <p>UNIT_TIME</p>

	Time (s).
UNIT_KV	Servo gain factor (1/s).
UNIT_REV_FEED	Rotary feed rate (mm/rev, inch/rev).
UNIT_INERTIA	Moment of inertia (kgm <sup>2</sup> )
UNIT_PERCENT	Percent (%).
UNIT_FREQU	Frequency (Hz)
UNIT_VOLTAGE	Voltage (V)
UNIT_CURRENT	Current (A)
UNIT_TEMP	Temperature (degrees Celsius)
UNIT_ANGLE	Angle (degrees)
UNIT_RESIST	Resistance (ohm)
UNIT_INDUCT	Inductance (mH)
UNIT_TORQUE	Torque (Nm)
UNIT_TORQUE_PER_CURR	Torque constant (Nm/A)
UNIT_CURR_CTRL_GAIN	Current controller gain (V/A)
UNIT_SPEED_CTRL_GAIN	Speed controller gain (Nm/rad s <sup>-1</sup> )
UNIT_ROT_SPEED	Speed (rpm)
UNIT_611D_TIME1	Time (31.25µs)
UNIT_611D_TIME2	Time (µs)
UNIT_611D_TIME3	

	Time (ms)
	UNIT_611D_TIME4
	Time (s)
	UNIT_POWER
	Power (kW)
	UNIT_CURRENT_SMALL
	Small current ( $\mu$ A)
	UNIT_611D_VS
	(Vs)
	UNIT_611D_VS_SMALL
	( $\mu$ Vs)
	UNIT_TORQUE_SMALL
	Small torque ( $\mu$ Nm)
	UNIT_611D_AOVERVS
	(A / Vs)
	UNIT_PROMILLE
	1/1000
	UNIT_HZ_PER_SEC
	Hz/s
<i>cursor_txt_id</i>	Text number for an operator info text which is displayed in the dialog line when the dialog cursor enters the field. (OP030 only).
<i>item_index_typ</i>	Based on the table item definition, several lines can be displayed in one column. The number is dependent on the value derived from the variable ( <i>v_item_index</i> ). Generation of the data identifier (address) for these lines is controlled by the type of indexing set ( <i>item_index_typ</i> ). For each address, part of the address is indexed. Which part of the address is incremented can be specified in this parameter ( <i>item_index_typ</i> ). Indexing starts with the base address and continues until base address + index, from the following indexing value ( <i>v_item_index</i> , ...p3).  You can choose from the following indexing types:
	VAR_TOKEN_INDEX_SECTION
	Index the unit (see also section on data identifiers for variable access).
	VAR_TOKEN_INDEX_ROW
	Index the variable address line (see also section on data identifiers for variable access).
	VAR_TOKEN_INDEX_COLUMN

Index the variable address column (see also section on data identifiers for variable access).

Example: Actual value for all geometry axes:

*v\_adr,..p3:*  
P\_C\_SGA\_progDistToGo, 1, 0, 0

*item\_index\_typ:*  
VAR\_TOKEN\_INDEX\_ROW

*v\_item\_index..p3:*  
P\_C\_Y\_numGeoAxes, 1, 0, 0

*v\_item\_index* *v\_item\_index\_p1, v\_item\_index\_p2, v\_item\_index\_p3.* Via this parameter you can directly or indirectly set the number of table lines to be expanded from this table item.

Specify a number directly (OP030 only):

**P\_VALUE, num\_lines, 0, 0**

To specify a number indirectly access an NC/PLC or MMC variable with a **data identifier** (see also section on data identifiers for variable access). The returned value is used as the number of table lines.

### Example

Configuring a table with three columns:

Column 1: Output a text which is defined in TAB\_ITEM txt\_id\_1.

Column 2: Output a value specified in TAB\_ITEM via v\_adr, v\_p1, v\_p2, v\_p3.

Column 3: Output a physical unit entered in TAB\_ITEM phys\_unit.

In the first TAB\_ITEM the number of lines is expanded to correspond to the number of axes in the channel. The variable P\_C\_SEMA\_lag is indexed in the unit with P\_C\_Y\_numMachAxes, see also section on data identifiers for variable access.

BEGIN\_OBJECT\_LIST (OB\_TAB\_ITEM\_SEMA)

```
TAB_ITEM ( 10,
           P_C_SEMA_lag, 1, 0, 0, /* token column identifier */
           CON_DECIMAL, F_DOUBLE, 0, CON_IO_RESOLUTION,
                                     /* convert function */
           CHECK_OFF, 0.0 , 0.0,      /* limit check */
           TAB_TXT_INDEX_1|TAB_TXT_BRACKETS, /* text attributes */
           0,                          /*access test */
           T_FOLLOWING_ERROR,          /* text-id */
           UNIT_POSN,                  /* physical unit */
           0,                          /* cursor text number only OP030 */
           VAR_TOKEN_INDEX_ROW,
           P_C_Y_numMachAxes, 1, 0, 0, )
```

```

TAB_ITEM ( 20,
  P_C_SEMA_trackErrContr, 1, 0, 0, /* token column identifier */
  CON_DECIMAL, F_DOUBLE, 0, CON_IO_RESOLUTION,
                                     /* convert function */
  CHECK_OFF, 0.0 , 0.0, /* limit check */
  0, /* text attributes */
  0, /*access test */
  T_CONTROL_DEVIATION, /* text-id */
  UNIT_POSN, /* physical unit */
  0, /* cursor text number only OP030 */
  0, /* index typ */
  0,0,0,0 /* index column identifier */
)
TAB_ITEM ( 30,
  P_C_SEMA_trackErrDiff, 1, 0, 0, /* token column identifier */
  CON_DECIMAL, F_DOUBLE, 0, CON_IO_RESOLUTION,
                                     /* convert function */
  CHECK_OFF, 0.0 , 0.0, /* limit check */
  0, /* text attributes */
  0, /*access test */
  T_CONTOUR_DEVIATION, /* text-id */
  UNIT_POSN, /* physical unit */
  0, /* cursor text number only OP030 */
  0, /* index type */
  0,0,0,0 /* index column identifier */
)

```

```
END_OBJECT_LIST (OB_TAB_ITEM_SEMA)
```

```
BEGIN_OBJECT_LIST (OB_TAB_COLUMN_SEMA)
```

```

  TAB_COLUMN (10, SEMA_TAB_X_NAME, SEMA_TAB_Y_ROW,
  SEMA_TAB_W_NAME, /* position of column */
  CS_ASIA_SMALL, /* set of characters */
  W_O_TCOL, W_O_FCOL, /* foreground color,
  background color */
  IO_LEFT_ADJUST, /* attributes */
  OP_OUTPUT_FIELD, /* type of column */
  TEST_RC_SEMA_NAME, /* refresh cycle */
  COL_1, /* column index */

```



```

        VAR_TEXT,                                /* column attributes */
        OBJECT_LIST_PTR (OB_TAB_ITEM_SEMA)
                                                /* ptr to TAB_ITEM */
    )
    TAB_COLUMN (20, SEMA_TAB_X_VALUE, SEMA_TAB_Y_ROW,
SEMA_TAB_W_VALUE,                            /* position of column */
    CS_SMALL,                                  /* set of characters */
    W_O_TCOL, W_O_FCOL,                       /* foreground color,
                                                background colour */
    0,                                          /* attributes */
    OP_OUTPUT_FIELD,                          /* type of column */
    TEST_RC_SEMA_VALUE,                      /* refresh cycle */
    COL_2,                                    /* column index */
    0,                                          /* column attributes */
    OBJECT_LIST_PTR (OB_TAB_ITEM_SEMA)
                                                /* ptr to TAB_ITEM */
    )
    TAB_COLUMN (30, SEMA_TAB_X_UNIT, SEMA_TAB_Y_ROW,
SEMA_TAB_W_UNIT                              /* position of column */
    CS_ASIA_SMALL,                            /* set of characters */
    W_O_TCOL, W_O_FCOL,                       /* foreground color,
                                                background color */
    IO_LEFT_ADJUST,                           /* attributes */
    OP_OUTPUT_FIELD,                          /* type of column */
    TEST_RC_SEMA_UNIT,                       /* refresh cycle */
    COL_3,                                    /* column index */
    VAR_UNIT,                                 /* column attributes */
    OBJECT_LIST_PTR (OB_TAB_ITEM_SEMA)
                                                /* ptr to TAB_ITEM */
    )
END_OBJECT_LIST (OB_TAB_COLUMN_SEMA)

BEGIN_OBJECT_LIST (OB_SEMA_TAB)
    DEF_SCROLL_BAR (10, TEST_SEMA_SB_ID, W_TEST_SEMA,
TEST_SEMA_X_SCR_BAR, TEST_SEMA_Y_SCR_BAR,
    TEST_SEMA_WIDTH_SCR_BAR,
    TEST_SEMA_HEIGHT_SCR_BAR,
    TEST_SEMA_SCR_BAR_FCOL, TEST_SEMA_SCR_BAR_BCOL,
    Y_DIRECTION, TEST_SEMA_RC_TAB,
    0, 0, 0, 0)
    TABLE (30000, TEST_X_TAB, TEST_Y_TAB, TEST_W_TAB,
TEST_H_TAB,                                /* position */

```

```

TEST_ROW_DIST_TAB, TEST_ROW_CNT_TAB,
/* row distance, row count */
W_FCOL, /* background color */
TEST_SEMA_SB_ID, /* scrollbar-id */
0, /* not used */
OBJECT_LIST_PTR (OB_TAB_COLUMN_SEMA),
/* ptr to tab columns */
NULL, /* ptr to graf.objectlist */
LIS_STANDARD, VAR_SERVER, /* id of server */
0, 0, 0, 0) /* server parameter */
END_OBJECT_LIST (OB_SEMA_TAB)

BEGIN_OBJECT_LIST (OB_TEST)
/* background */
RECTANGLE ( 1000, BEGIN_X, BEGIN_Y, WIDTH_M_INI,
HEIGHT_M_INI, FILLED, W_FCOL, 0xff )
/* header */
RECTANGLE ( 1010, BEGIN_X, BEGIN_Y, WIDTH_M_INI, HEADER,
FILLED, W_HL_FCOL, 0xff )
/* Table */
MACRO ( 1020, TEST_X_TAB, TEST_Y_TAB, OBJECT_LIST_PTR
(OB_SEMA_TAB) )
END_OBJECT_LIST (OB_TEST)

```

### 3.2.13 Graphic list field - PICT\_FIELD

<b>Description</b>	A static object list is output once or cyclically in a graphic list field, controlled by the value of a linked variable from the NCK/PLC or MMC.
<b>Syntax</b>	<b>PICT_FIELD</b> ( <i>id, x, y, w, h, bc, refresh, v_adr, v_p1, v_p2, v_p3</i> )
<b>Parameters</b>	<p><i>id</i> Unique identifier for the PICT_FIELD object in the current module.</p> <p><i>x, y</i> Basis position of graphic. Top left-hand corner of field in pixels relative to the higher-level object.</p> <p><i>w, h</i> Width and height of the field in pixels. When refreshed, this area is always cleared in the background color of the field.</p>

<i>bc</i>	Background color. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for applicable values).
<i>refresh</i>	Refresh cycle for visualization (display refresh) and data access (data refresh). Also refer to Section "Refresh factor".
<i>v_adr</i>	Data identifier for accessing NC/PLC/MMC data (also refer to section on data identifier) <b>This data access must return the identifier for the static object list (<i>obl_id</i>) to be displayed.</b>
<i>v_p1, v_p2, v_p3</i>	Additional parameters for data access.

**Example**

Often used to output different objects lists (controlled via an MMC notepad entry); in this case, each alternatively displayed parameter is output in a separate object list.

```
/* Define different object lists with different polymarkers*/
```

```
BEGIN_OBJECT_LIST (101)
POLYMARKER (2, 0, 0, POLY_RESET, WHITE)
END_OBJECT_LIST (101)
```

```
BEGIN_OBJECT_LIST (102)
POLYMARKER (3, 0, 0, POLY_CANCEL, WHITE)
END_OBJECT_LIST (102)
```

```
BEGIN_OBJECT_LIST (103)
POLYMARKER (5, 0, 0, POLY_NC_START, WHITE)
END_OBJECT_LIST (103)
```

```
* Define object list with graphic list field */
```

```
BEGIN_OBJECT_LIST(200)
```

```
/* Output the object list set in notepad entry 25.
```

In the example, the object list with ID **101** is set and the polymarker it contains is displayed.

```
*/
```

```
PICT_FIELD (10, 20, 20, 16, 16
BLACK,
0,
P_NB, 25, 0, 0)
```

```
END_OBJECT_LIST(200)
```

```
BEGIN_OPEN_LIST (201)
```

```
/* Save object list ID 101 in notepad entry 25 */
```

```
AC_SET_WORD (211, 101, P_NB, 25, 0, 0)
```

```
END_OPEN_LIST (201)
```

### 3.2.14 Action field - ACTION\_FIELD

<b>Description</b>	<p>These fields do not display values of the control system, but trigger, in response to events and in conjunction with reaction routine D_ACTIVATE_ACTION, the actions stored in the assigned action list (<i>act_li_id</i>).</p> <p>An action field is not visible on the screen. It does not become visible until the dialog cursor which inverts the appropriate screen area is placed over it. In other words, the action field must be positioned on a text, which defines its function (action).</p> <p>For the action list specified by the action field on which the cursor is positioned to be executed, a defined event must occur. This event must be defined via a reaction element in the reaction list of the window to which the object list containing the action field belongs.</p>														
<b>Syntax</b>	<pre><b>ACTION_FIELD</b> (<i>id</i>, <i>x</i>, <i>y</i>, <i>w</i>, <i>h</i>,                 <i>field_attr</i>[<i>field_attr</i>...]                 <i>cursor_txt_id</i>,                 <i>cur_r</i>, <i>cur_l</i>, <i>cur_d</i>, <i>cur_u</i>,                 <b>ACTION_LIST_PTR</b> (<i>acl_id</i>))</pre>														
<b>Parameters</b>	<table border="0"> <tr> <td style="vertical-align: top;"><i>id</i></td> <td>Unique identifier for the ACTION_FIELD object in the current module.</td> </tr> <tr> <td style="vertical-align: top;"><i>x</i>, <i>y</i></td> <td>Position of top left-hand corner of field in pixels relative to the object above it.</td> </tr> <tr> <td style="vertical-align: top;"><i>w</i>, <i>h</i></td> <td>Width and height of the field in pixels.</td> </tr> <tr> <td style="vertical-align: top;"><i>field_attr</i></td> <td>Attribute parameter: IO_CURSOR_DISABLE Field is skipped by the dialog cursor.</td> </tr> <tr> <td style="vertical-align: top;"><i>cursor_txt_id</i></td> <td>Text number for an operator info text which is displayed in the dialog line when the dialog cursor enters the field.</td> </tr> <tr> <td style="vertical-align: top;"><i>cur_r</i>, <i>cur_l</i>, <i>cur_d</i>, <i>cur_u</i>,</td> <td>Identifiers of the fields where the dialog cursor for cursor right, cursor left or cursor down, cursor up changes (KEY_RIGHT, KEY_LEFT, KEY_DOWN, KEY_UP).</td> </tr> <tr> <td style="vertical-align: top;"><i>acl_id</i></td> <td>Unique identifier of an action list which is executed via reaction D_ACTIVATE_ACTION when the appropriate event occurs.</td> </tr> </table>	<i>id</i>	Unique identifier for the ACTION_FIELD object in the current module.	<i>x</i> , <i>y</i>	Position of top left-hand corner of field in pixels relative to the object above it.	<i>w</i> , <i>h</i>	Width and height of the field in pixels.	<i>field_attr</i>	Attribute parameter: IO_CURSOR_DISABLE Field is skipped by the dialog cursor.	<i>cursor_txt_id</i>	Text number for an operator info text which is displayed in the dialog line when the dialog cursor enters the field.	<i>cur_r</i> , <i>cur_l</i> , <i>cur_d</i> , <i>cur_u</i> ,	Identifiers of the fields where the dialog cursor for cursor right, cursor left or cursor down, cursor up changes (KEY_RIGHT, KEY_LEFT, KEY_DOWN, KEY_UP).	<i>acl_id</i>	Unique identifier of an action list which is executed via reaction D_ACTIVATE_ACTION when the appropriate event occurs.
<i>id</i>	Unique identifier for the ACTION_FIELD object in the current module.														
<i>x</i> , <i>y</i>	Position of top left-hand corner of field in pixels relative to the object above it.														
<i>w</i> , <i>h</i>	Width and height of the field in pixels.														
<i>field_attr</i>	Attribute parameter: IO_CURSOR_DISABLE Field is skipped by the dialog cursor.														
<i>cursor_txt_id</i>	Text number for an operator info text which is displayed in the dialog line when the dialog cursor enters the field.														
<i>cur_r</i> , <i>cur_l</i> , <i>cur_d</i> , <i>cur_u</i> ,	Identifiers of the fields where the dialog cursor for cursor right, cursor left or cursor down, cursor up changes (KEY_RIGHT, KEY_LEFT, KEY_DOWN, KEY_UP).														
<i>acl_id</i>	Unique identifier of an action list which is executed via reaction D_ACTIVATE_ACTION when the appropriate event occurs.														
<b>Example</b>	<pre>BEGIN_ACTION_LIST (40002)     AC_SET_WORD ( 2, 10, P_NB, 0, 0, 0)     AC_..... END_ACTION_LIST (40002)  BEGIN_REACTION_LIST (40040)     <b>RC_D_ACTIVATE_ACTION (20, KEY_ENTER)</b></pre>														

```

...
END_REACTION_LIST (40040)

BEGIN_OBJECT_LIST(45020)
...
ACTION_FIELD
(2, 40, 60,      /* ID (here: 2), x, y */
50, 18, 0, 0,   /* width, height, attribute */
0,             /* CURSOR-TEXT_ID*/
6, 8, 10, 0,   /* Cursor sequences */
ACTION_LIST_PTR (40002)) /* pointer to the action list */
END_OBJECT_LIST(45020)
BEGIN_WINDOW (45010)
NULL,
NULL,
OBJECT_LIST_PTR (45020), /* Pointer to the object list */
REACTION_LIST_PTR (40040), /* Pointer to the response list */
NULL, NULL
END_WINDOW (45010)

```

Before action list **40002** can be executed, the dialog cursor must be positioned on the action field with identifier (ID) 2 and an event with code KEY\_ENTER must occur.

### 3.2.15 Inverse field - INVERSE\_FIELD

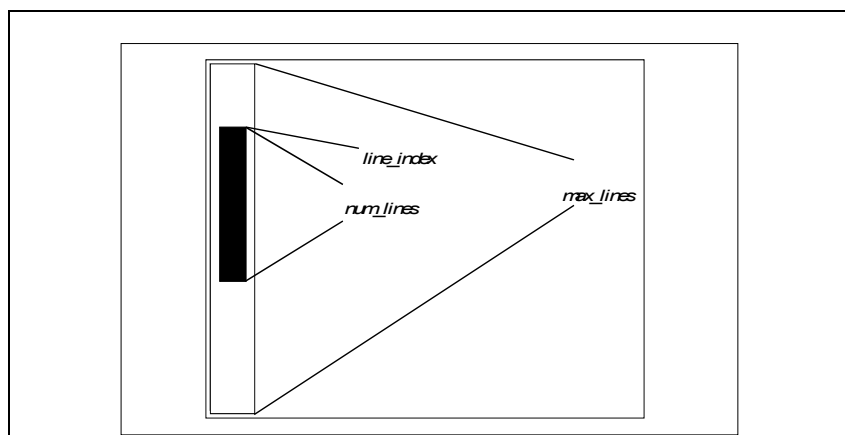
<b>Description</b>	<p>The purpose of an inverse field is to display a screen area either normally or in reverse video as a function of an NC/PLC/MMC variable.</p> <p>The field is displayed in reverse video when an NCK/PLC/MMC variable value (<i>v_adr</i>), that is ANDed with a bit mask (<i>bit_mask</i>), returns <b>1</b> or, with INV_ZERO_ACTIV, <b>0</b>.</p>																
<b>Syntax</b>	<pre>INVERSE_FIELD(<i>id</i>, <i>x</i>, <i>y</i>, <i>w</i>, <i>h</i>,,               <i>refresh</i>,               <i>attr</i>[<i>attr</i>...],               <i>bit_mask</i>,               <i>v_adr</i>, <i>v_p1</i>, <i>v_p2</i>, <i>v_p3</i>)</pre>																
<b>Parameters</b>	<table border="0"> <tr> <td style="vertical-align: top;"><i>id</i></td> <td>Unique identifier for the INVERSE_FIELD object in the current module.</td> </tr> <tr> <td style="vertical-align: top;"><i>x</i>, <i>y</i></td> <td>Position of top left-hand corner of field in pixels relative to the object above it.</td> </tr> <tr> <td style="vertical-align: top;"><i>w</i>, <i>h</i></td> <td>Width and height of the field in pixels.</td> </tr> <tr> <td style="vertical-align: top;"><i>refresh</i></td> <td>Refresh cycle for visualization (display refresh) and data access (data refresh). (see also Section "Refresh factor").</td> </tr> <tr> <td style="vertical-align: top;"><i>attr</i></td> <td>Attribute for the inverse field.  INV_ZERO_ACTIVE The inverse field is low-active, i.e. displayed in  INV_BLINK The field flashes when it is switched to reverse video.</td> </tr> <tr> <td style="vertical-align: top;"><i>bit_mask</i></td> <td>Bit mask which is ANDed with the value supplied under <i>v_adr</i>. The length of the bit mask is 1 byte.</td> </tr> <tr> <td style="vertical-align: top;"><i>v_adr</i></td> <td>Data identifier for accessing NC/PLC/MMC data.</td> </tr> <tr> <td style="vertical-align: top;"><i>v_p1</i>, <i>v_p2</i>, <i>v_p3</i></td> <td>Additional parameters for data access.</td> </tr> </table>	<i>id</i>	Unique identifier for the INVERSE_FIELD object in the current module.	<i>x</i> , <i>y</i>	Position of top left-hand corner of field in pixels relative to the object above it.	<i>w</i> , <i>h</i>	Width and height of the field in pixels.	<i>refresh</i>	Refresh cycle for visualization (display refresh) and data access (data refresh). (see also Section "Refresh factor").	<i>attr</i>	Attribute for the inverse field.  INV_ZERO_ACTIVE The inverse field is low-active, i.e. displayed in  INV_BLINK The field flashes when it is switched to reverse video.	<i>bit_mask</i>	Bit mask which is ANDed with the value supplied under <i>v_adr</i> . The length of the bit mask is 1 byte.	<i>v_adr</i>	Data identifier for accessing NC/PLC/MMC data.	<i>v_p1</i> , <i>v_p2</i> , <i>v_p3</i>	Additional parameters for data access.
<i>id</i>	Unique identifier for the INVERSE_FIELD object in the current module.																
<i>x</i> , <i>y</i>	Position of top left-hand corner of field in pixels relative to the object above it.																
<i>w</i> , <i>h</i>	Width and height of the field in pixels.																
<i>refresh</i>	Refresh cycle for visualization (display refresh) and data access (data refresh). (see also Section "Refresh factor").																
<i>attr</i>	Attribute for the inverse field.  INV_ZERO_ACTIVE The inverse field is low-active, i.e. displayed in  INV_BLINK The field flashes when it is switched to reverse video.																
<i>bit_mask</i>	Bit mask which is ANDed with the value supplied under <i>v_adr</i> . The length of the bit mask is 1 byte.																
<i>v_adr</i>	Data identifier for accessing NC/PLC/MMC data.																
<i>v_p1</i> , <i>v_p2</i> , <i>v_p3</i>	Additional parameters for data access.																

### 3.2.16 Scrollbar - DEF\_SCROLL\_BAR

<b>Description</b>	<p>The purpose of a scrollbar is to indicate that there are more data available than can be displayed in the current window.</p> <p>The bar represents 100% of the available data. The slide control (box) indicates the current position in the available data and the relation between the data displayed in the window and the total data quantity.</p> <p>Normally, you specify the value for (<i>line_index</i>) using a notepad entry. The new status is displayed when the value of the notepad entry content is changed (in, for example, a reaction list as a reaction to KEY_PAGE_MINUS,</p>
--------------------	--

KEY\_PAGE\_PLUS, KEY\_UP, KEY\_DOWN, ..) and a scrollbar refresh using reaction routine RC\_SCROLL\_BAR\_REFRESH is called.

The scrollbar is automatically refreshed with the Table and Editor control elements. The link to these is made via the scrollbar index (*sb\_nr*). You do not need to call RC\_SCROLL\_BAR\_REFRESH when you configure a table (TABLE).



### Syntax

```
DEF_SCROLL_BAR (id,
                sb_nr, win_id,
                x, y, w, h,
                fc, bc,
                scroll_dir,
                num_lines,
                nb_nr_max_lines,
                max_lines,
                nb_nr_line_index,
                line_index)
```

### Parameters

<i>id</i>	Unique identifier for the SCROLL_BAR object in the current module.
<i>win_id</i>	Identifier of the window to which the scrollbar belongs 0 Scrollbar belongs to the current window.
<i>sb_nr</i>	Scrollbar index. This index must be between 1 and 9. A total of 10 scrollbars may be used simultaneously. The index is also used in conjunction with tables and the editor object.
<i>x</i> , <i>y</i>	Position of top left-hand corner of scrollbar in pixels relative to the object above it.
<i>w</i> , <i>h</i>	Width and height of the scrollbar in pixels.
<i>fc</i>	Foreground color. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
<i>bc</i>	Background color. Colors and gray scales are system-dependent (refer to the Section <b>Colors</b> for valid values).
<i>scroll_dir</i>	Scroll direction:

	Y_DIRECTION (currently possible value).
<i>num_lines</i>	Number of lines shown in the display (line cutout).
<i>nb_nr_max_lines</i>	Notepad entry identifier: NOTEPAD if the following parameter is a notepad entry number. 0 if the following parameter is directly used as value.
<i>max_lines</i>	Total number of lines (100%). Sum of lines calculated in the line cutout ( <i>num_lines</i> ) and line index ( <i>line_index</i> ).
<i>nb_nr_line_index</i>	Notepad entry identifier: NOTEPAD if the following parameter is a notepad entry number. 0 if the following parameter is directly used as value.
<i>line_index</i>	Line index for the first line on the screen ( <i>num_lines</i> ) referred to the total number of lines ( <i>max_lines</i> ).

### 3.2.17 Macro element (sub-object lists) - MACRO

<b>Description</b>	<p>A macro element is a collection of individual objects and other macro elements within object lists. This results in nesting (max. nesting depth is 10 macros) for display building purposes.</p> <p>The positions of the objects contained in a macro element refer to the coordinate system of the macro element which has just been processed (i.e. always relative to the position of the last macro). The position of the highest macro element refers in turn to the zero point of the window or to the higher-level macro element.</p> <p>Like subroutines, macros have the advantage that they can be used at different locations.</p>
<b>Syntax</b>	<b>MACRO</b> ( <i>id</i> , <i>x</i> , <i>y</i> , OBJECT_LIST_PTR ( <i>obl_id</i> ))
<b>Parameters</b>	<i>id</i> Unique identifier for the MACRO object in the current module.



<i>x, y</i>	Position of top left-hand corner of macro in pixels relative to the object above it.
<i>obl_id</i>	Unique identifier of an object list to be displayed relative to the specified macro positions.

### 3.2.18 Dynamic macro element (sub-object lists) - MACRO\_DYN

**Description** A macro element is a collection of individual objects and other macro elements within object lists. This results in nesting (max. nesting depth is 10 macros) for display building purposes.

The positions of the objects contained in a macro element refer to the coordinate system of the macro element which has just been processed (i.e. always relative to the position of the last macro). The position of the highest macro element refers in turn to the zero point of the window or to the higher-level macro element.

Like subroutines, macros have the advantage that they can be used at different locations.

**Syntax** **MACRO\_DYN** (*id, nb\_x, nb\_y, OBJECT\_LIST\_PTR (obl\_id)*)

**Parameters**

<i>id</i>	Unique identifier for the MACRO object in the current module.
<i>nb_x, nb_y</i>	Numbers of notepads containing the position of the top, left-hand corner of the macro in pixels relative to the object above.
<i>obl_id</i>	Unique identifier of an object list to be displayed relative to the specified macro positions.

### 3.2.19 Progress bar - PROGRESS\_BAR (MMC100/EBF)

**Description** This element is for visualizing a value (not a string). This is a pure output field, i.e. it cannot contain a cursor. It has a minimum and a maximum value (display range) plus a signal value (100% value) above which the bar is displayed in a configurable signal color.

**Syntax** **PROGRESS\_BAR**(*id, x, y, w, h, fc, bc, sc, a, ac, rc, min, max, sig, v\_adr, v\_p1, v\_p2, v\_p3*)

**Parameters**

<i>id</i>	Unique identifier of field within the module. This ID must be different to the IDs of other dynamic fields contained in the module since it is treated in the same way as a normal O_FIELD.
-----------	---

<i>x, y</i>	x, y position relative to window or the macro located above
<i>w, h</i>	Width, height of the complete output range
<i>fc, bc,</i>	Foreground color (color of the value display), background color
<i>sc</i>	Signal color (100% value), color of display when signal value is exceeded.
<i>a</i>	Attribute with the following meaning PBAR_VERTICAL Displayed in vertical direction PBAR_BACKWARD Displayed from right to left or from top to bottom PBAR_SIGNAL_COL Display above the signal value in the signal color
<i>ac</i>	Access Level
<i>rc</i>	Refresh
<i>Min, max</i>	Minimum value, maximum value in DOUBLE
<i>sig</i>	Signal value
<i>v_adr</i>	Data identifier for accessing NC/PLC/MMC data.
<i>v_p1, v_p2, v_p3</i>	Additional parameters for data access.

### 3.2.20 Tachometer element - TACHO (MMC100/EBF)

<b>Description</b>	This element is for visualizing a value (not a string). It is purely an output field, i.e. it cannot be manipulated with the cursor. It has a minimum and a maximum value (display range) plus a signal value (100% value) above which the bar is displayed in a configurable signal color. The value is displayed in the form of a vector.								
<b>Syntax</b>	<b>TACHO</b> ( <i>id, x, y, r, fw, fc, bc, sc, a, ac, rc, min, max, sig, v_adr, v_p1, v_p2, v_p3</i> )								
<b>Parameters</b>	<table> <tr> <td><i>id</i></td> <td>Unique identifier of field within the module. This ID must be different to the IDs of other dynamic fields contained in the module since it is treated in the same way as a normal O_FIELD.</td> </tr> <tr> <td><i>x, y</i></td> <td>x, y position relative to the Window or the macro located above it</td> </tr> <tr> <td><i>r</i></td> <td>Radius</td> </tr> <tr> <td><i>fw</i></td> <td>always 0</td> </tr> </table>	<i>id</i>	Unique identifier of field within the module. This ID must be different to the IDs of other dynamic fields contained in the module since it is treated in the same way as a normal O_FIELD.	<i>x, y</i>	x, y position relative to the Window or the macro located above it	<i>r</i>	Radius	<i>fw</i>	always 0
<i>id</i>	Unique identifier of field within the module. This ID must be different to the IDs of other dynamic fields contained in the module since it is treated in the same way as a normal O_FIELD.								
<i>x, y</i>	x, y position relative to the Window or the macro located above it								
<i>r</i>	Radius								
<i>fw</i>	always 0								

<i>fc, bc,</i>	Foreground color (color of the value display), background color
<i>sc</i>	Signal color (100% value), color of display when signal value is exceeded.
<i>a</i>	Attribute with the following meaning TACHO_RIGHT_START Display from the right to the left TACHO_SIGNAL_COL Pointer color above the signal value in the signal color
<i>ac</i>	Access Level
<i>rc</i>	Refresh
<i>Min, max</i>	Minimum value, maximum value in DOUBLE
<i>sig</i>	Signal value
<i>v_adr</i>	Data identifier for accessing NC/PLC/MMC data.
<i>v_p1, v_p2, v_p3</i>	Additional parameters for data access.

### 3.2.21 Bitmaps - PCX (MMC100/EBF)

<b>Description</b>	<p>MMC 100/UOP offers the option to display bitmaps. These images are created with any appropriate tool and stored in PCX format with 16 colors. Images in this format are converted with the PCXCONV tool on the application disk and then compressed with ARJ. The files created by the conversion have the extension <b>.BIN</b>.</p> <p>The image is output in its full size at the specified position.</p> <p>Version 06.02 and higher supports conversion of 16-color bitmaps (.bmp) directly to binary format by means of the 'bmp2bin.exe' tool included in the scope of delivery. If you invoke the tool without parameters, it will display the required parameters.</p> <p>Call, e.g. bmp2bin e:\my_dir*\*.bmp -&gt; all .bmp files that reside in the specified directory are converted and saved to the same directory.</p>
<b>Syntax</b>	<b>PCX</b> (id, x, y, arj_name, pcx_name)
<b>Parameters</b>	<p><i>id</i> Unique identifier of field within the module. This ID must be different to the IDs of other dynamic fields contained in the module.</p> <p><i>x, y</i> Positions in pixels relative to the element above (window, macro).</p> <p><i>arj_name</i> Name of archive file in which converted image is stored (in inverted commas). A maximum of 8 characters + 3 characters extension are permitted (e.g. "gp_help.arj").</p>

*pcx\_name* Name of converted file (in inverted commas). You can use up to 8 characters, (e.g. "fci\_sp" without extension)

**Note**

PCX images must be converted to binary data using the `pcxconv.exe` tool and then stored in files so that they can be interpreted by the MMC 100.

These images must have a width and an x position which is integrally divisible by 8.

They are stored in a format which can be displayed only on 640 \* 480 pixel VGAs with 16 colors.

Call:

**`pcxconv x y w h file bin_len colTab`**

Meaning of parameters:

<i>x</i>	Absolute screen position from which the image is read in (must be divisible by 8)
<i>y</i>	Absolute screen position from which the image is read in.
<i>w</i>	Width in pixels (must be an integer multiple of 8)
<i>h</i>	Height in pixels
<i>file</i>	PCX file to be converted including path (wildcards can also be specified (e.g. *.pcx))
<i>bin_len</i>	Maximum length of binary files to be generated (0 means: Only files with max. 10KB are generated, 1 means: Only files with max. 26KB, 2 means unrestricted file length). The recommended setting here is 2!
<i>colTab</i>	Color table to be used (default: MMC 100/UOP, cmm: ShopMill, ManTurn)

Batch file

...mmc100pj\instutil\conv\_pcx.bat

can be used as an example for creation and packing.

Use the installation kit to copy the packed file (Copy external Files into Project) to the target system.

## 3.3 Colors

Numerical values are used to define colors. OP030, HPU and MMC 100/UOP use different color definitions.

### 3.3.1 Colors for OP 030 and HPU

The **OP030** and **HPU** only use color definitions **BLACK** and **WHITE**, i.e. only **BLACK** or **WHITE** may be specified for the parameter *color* in the fields of the graphic element.

### 3.3.2 Colors for MMC100/UOP

The screen for the MMC 100/UOP is available in either color or monochrome. To ensure that the same configuration can be used for both variants, you must take into account both the grayscale shading and color information when defining colors.

The color information parameter for a graphic element consists of a 16-bit data word in which the color information is stored in the 8 higher-order bits and the grayscale information in the 8 lower-order bits.

The following grayscale values are defined (becoming lighter as grey number increases):

- **BLACK**
- **GREY\_0**
- **GREY\_1**
- **GREY\_2**
- **GREY\_3**
- **GREY\_4**
- **GREY\_5**
- **GREY\_6**
- **WHITE\_M**
- **NO\_COLOR** --> no color

The following color values are defined:

- **BLACK**
- **BLACK\_GREY**
- **BLACK\_W**
- **DARK\_GREY**
- **HL\_GREY**
- **LIGHT\_GREY**
- **WHITE\_C**
- **WHITE\_2**

- **WHITE3**
- **WHITE**
- **ORANGE**
- **ORANGE\_D**
- **PETROL**
- **BLUE**
- **RED**
- **YELLOW**
- **NO\_COLOR** --> no color

To simplify the configuring of colors and gray scales, you can create bit combinations of one color shade and one grey shade in each case.

For example, you can store the following combination in a header file:

```
/* Background color for alarm line */  
#define GM_AL_FCOL    (WHITE3 | WHITE_M)  
/* Text color for alarms */  
#define GM_AL_TCOL    (RED | BLACK)  
/* Text color for program comments */  
#define GM_KOM_TCOL   (PETROL | BLACK)  
etc.  
/* General background color for IO fields */  
#define IO_BC_COL     (YELLOW | WHITE_M)
```

## 3.4 Refresh factor – display and data refresh

### Description

The refresh cycle specifies both a factor for the visualization refresh rate (display refresh) and for the data access refresh rate (data refresh).

#### Display refresh:

*refresh* specifies a factor representing a multiple of 100 ms in which the field must be refreshed on the screen.

#### Data refresh after change:

*refresh* equals **0**:

Single data access; data is read exactly once when field is displayed.

*refresh* equals **1**:

Cyclical data refresh **only when value is changed**, but max. every **100 ms**

*refresh* greater than **1 and less than 10**:

Cyclical data refresh **only when value is changed**, but max. every **300 ms**

*refresh* greater than or equal to **10** and less than **255**:

Cyclical data refresh **only when value is changed**, but max. every **1 s**

#### Change-independent data refresh in a fixed time frame:

If *refresh* is combined with attribute REFRESH\_PERMANENT (e.g. "1|REFRESH\_PERMANENT"), then data is refreshed in exactly the specified time frame irrespective of any data change.

When *refresh* equals 0, this parameter is not permissible. When *refresh* is set to any other value, the same classification applies to the data refresh time as to data refresh in response to a change.

---

#### Note

- You should be aware that change-independent cyclical data accessing places the greatest burden on the whole system in terms of its performance.
  - The data accessing service in response to changes is supported for NCK variables only, i.e. not for PLC and MMC variables. You may still, however, specify the parameter.
- 

We recommend that you use existing preprocessor defines for the different refresh classes:

```
#define REFR_SINGLE 0
#define REFR_SLOW 10
#define REFR_MEDIUM 5
#define REFR_FAST 1
```

### Note

The more frequently your display and data are refreshed, the more computing power will be needed for the display.

When cyclical refresh is selected, the variable address is not regenerated each time. In other words, an error will occur if the display is not re-build after the index of a variable has been changed.

## 3.5 Data conversion

### Description

Conversion routines are provided with which you can convert NC/PLC/MMC variables into a display format (ASCII) and from the display/input format into an internal format (byte, word, floating point). You specify the conversion method when configuring dynamic fields.

You cannot convert all internal formats into all display formats - for further details, see the end of this chapter.

**Variables values from the NC, PLC or the MMC are assigned to dynamic display fields either**

- directly (bit, BYTE, (U)WORD, (U)LONG, DOUBLE and 8 single characters),
- as a reference (text number) or
- as a pointer to a string.

These entries can be displayed in different ways on the interface by the various conversion routines:

Screen mode	Variable content	Identifier for file conversion
Fixed text from a	Text number	CON_TEXT
Text file	Text number offset	CON_TEXT_OFFSET
	0, or value other than 0	CON_TEXT_BOOL
String	8 single ASCII characters	CON_ASCII
Dynamic	Pointer to string	CON_STRING
	Pointer to string	CON_STRING_LIMIT
Decimal value	Byte, unsigned byte, word, unsigned word, long, unsigned long, float, double	CON_DECIMAL
Hexadecimal value	Unsigned byte, unsigned word, unsigned long	CON_HEX
Binary value - bit pattern	Unsigned byte, unsigned word, unsigned long	CON_BINARY
BCD representation	unsigned Long	CON_BCD
Boolean display value dependent on bits in value.	Byte, unsigned byte, word, unsigned word, long, unsigned long	CON_BIT
No conversion	-	CON_OFF

You can parameterize the individual conversion methods specifically by setting the conversion parameters.



### 3.5.1 Data format for the conversion

Dynamic fields are linked to the data conversion via an 8-byte wide area. These eight bytes are interpreted differently depending on the conversion routine and setting of the data format parameter below. The data formats are explained in general here. How they are used is explained in the description of the different conversion routines.

F_BYTE	Signed 8-bit value. Only the 8 low-order bits of the read value are displayed in each case.
F_UBYTE:	Unsigned 8-bit value. Only the 8 low-order bits of the read value are displayed in each case.
F_WORD:	Signed 16-bit value. Only the 16 low-order bits of the read value are displayed in each case.
F_UWORD:	Unsigned 16-bit value. Only the 16 low-order bits of the read value are displayed in each case.
F_LONG:	Signed 32-bit value. Only the 32 low-order bits of the read value are displayed in each case.
F_ULONG:	Unsigned 32-bit value. Only the 32 low-order bits of the read value are displayed in each case.
F_FLOAT:	32-bit floating-point value. Only the 32 low-order bits of the read value are displayed in each case.
F_DOUBLE:	64-bit floating-point value.
F_POINTER:	32-bit pointer. Only the 32 low-order bits of the read value are used for conversion.

### 3.5.2 CON\_TEXT

**Description** This conversion is used to display a fixed text contained in a configuring text file.

**References:** /FB0/, SINUMERIK 840D/810D/FM-NC  
Configuring OP 030 Operator Interface  
/PK/, SINUMERIK MMC100/UOP Configuring Package

The value supplied to the dynamic fields is interpreted as a text number.

The conversion parameters have no meaning.

**Syntax** **CON\_TEXT;**  
**0, 0, 0**

### 3.5.3 CON\_TEXT\_OFFSET

**Description** This conversion method is used to display a variety of fixed texts depending on the value supplied.

The value supplied to the dynamic fields is interpreted as an offset to the text number specified in the conversion parameters. This text number plus the offset is then used to access the configuring text file.

**Syntax** CON\_TEXT\_OFFSET,  
*txt\_id\_basis*, 0, *max\_offset*

**Parameters**

*txt\_id\_basis* Basic text number. The addition of the basic text number (*txt\_id\_basis*) + read value produces the actual text number for accessing the text.

**The texts to be displayed must be stored sequentially in the configuring text file. The text with the basic text number must be the first text in the sequence.**

*max\_offset* A check is made to ascertain whether the read value exceeds *max\_offset*. In this case, only a question mark is output.

No check is made if *max\_offset* = 0, must ensure that texts with the calculated text number are available! Preset defines are stored in file MAX\_OFFS.H.

**Example**

Configuring text file:

```
TXT_NOT_IN_POS      "Not in position"
TXT_PLUS            "Travel direction, plus"
TXT_MINUS          "Travel direction, minus"
```

Values for variable P\_C\_SGA\_status (axis status):

```
0      Axis not in position
1      Travel command, plus direction
2      Travel command, minus direction
```

Output field definition:

```
O_FIELD (46, X_DIFF_ISTW,
        Y_DIFF_ISTW+HEIGHT_FELD_ISTW, ANZ_DIFF_ISTW,
        WD_TCOL, WD_FCOL,
        CHAR_SET,
        0,
        REFR_ISTWERT,
        P_C_SGA_status, NB_MA_ACT_GAXZ, NB_LINE, 2,
        CON_TEXT_OFFSET, TXT_NOT_IN_POS, 0, 0)
```

If the value for "Axis not in position" (0) is supplied, then text 'TXT\_NOT\_IN\_POS' is output.

If the value for "Travel command plus direction" (1) is supplied, then "1" is added to the basic text number and text 'TEXT\_PLUS' is output.

If the value for "Travel command minus direction" (2) is supplied, then "2" is added to the basic text number and text 'TEXT\_MINUS' is output.

### 3.5.4 CON\_TEXT\_BOOL

<b>Description</b>	Output exactly two different texts, dependent on the return value. If a value other than 0 is supplied to the dynamic field, then the text specified in the conversion parameter is output.  Otherwise (i.e. value equals 0), the text with text number + 1 is output.
<b>Syntax</b>	<b>CON_TEXT_BOOL,</b> <i>txt_id_l</i> , 0, 0
<b>Parameters</b>	<i>txt_id_l</i> Text number for a text in the configuring file. Text number where the <b>value is not 0</b> .  If the <b>value is 0</b> , then the text with <i>txt_id_l</i> +1 is displayed.
<b>Note</b>	The conversion method can be used only for read access operations.

### 3.5.5 CON\_ASCII

<b>Description</b>	The read value is interpreted and displayed byte for byte as ASCII characters. Exactly eight bytes are output. The conversion parameters have no meaning..
<b>Syntax</b>	<b>CON_ASCII,</b> 0,0,0
<b>Note</b>	The content of a variable is output byte for byte; any changes to the byte sequence caused by the Intel Little Endian Format for F_(U)WORD, F_(U)LONG are not reversed.  The character set is ANSI-compatible and available complete for the five language variants provided. The set includes the most commonly used characters, but not characters for special functions (e.g. scientific functions).

### 3.5.6 CON\_STRING

**Description** This conversion routine is used to display and input strings.

The read value is interpreted as a **pointer** to a string. The character string must end with '\0'. It is output in the dynamic display field.

This routine is predominantly used in connection with variables from the NCK area or predefined variables from the MMC area for output and input/output fields.

You can use the CON\_STRING routine only in conjunction with text variables for notepad variables (see example).

**Syntax** **CON\_STRING,**  
**{0 | CON\_DOS\_FILE | CON\_DOS\_PATH}, 0, 0**

**Parameters** 0 The character string is output without any further conversion.

CON\_DOS\_FILE NCK domain name (*\_N\_name\_typ*) is converted, in the display, into a file name in conformance with DOS (*name.typ*).

CON\_DOS\_PATH NCK domain name with prefix path (*/\_N\_pfad/\_N\_name\_typ*) is converted into a filename in conformance with DOS with path (*\pfad\name.typ*).

**Note** Parameters CON\_DOS\_FILE and CON\_DOS\_PATH can be set for read access operations only.

**Example** CON\_STRING in conjunction with text variables enables you to input and display strings:

```
BEGIN_OPEN_LIST( OP_M_APP )
....
/* Assign text variable 0 to notepads NB_20
and NB_21 */

AC_SET_TXT_NB (100, 0L, NB_20, 0)
....
END_OPEN_LIST( OP_M_APP )

BEGIN_OBJECT_LIST( OB_APP )
....
/* Display and input option for text variable 1 */
```

```

IO_FIELD (213, X_T_APP+150, Y_T_APP, 5,
          WHITE, BLACK,
          CS_SMALL,
          IO_LEFT_ADJUST,
          201, 214, 214, 214,
          0,
          REFR_MEDIUM,
          0,
          P_NB, NB_20, 0, 0,
          CON_STRING, 0, 0, 0 )

/* Display and input option for text variable 1 */
IO_FIELD (214, X_T_APP+150, Y_T_APP+30, 5,
          WHITE, BLACK,
          CS_SMALL,
          IO_LEFT_ADJUST,
          213, 213, 213, 213,
          0,
          REFR_MEDIUM,
          0,
          P_NB, NB_21, 0, 0,
          CON_STRING, 0, 0, 0 ) ...
END_OBJECT_LIST( OB_APP )

```

**Additional attribute** Attribute CON\_FORCE\_STRING is available additionally for data in the CON\_STRING format. When a string beginning with "\" is read, this attribute ensures that the field remains a string field, rather than being converted to a bitmap field.

### 3.5.7 CON\_STRING\_LIMIT

**Description** This conversion routine is used to display and input strings. If a string has to be truncated for display due to the field length restriction, this routine, as a special feature, marks it with an '\*'. Par3 is interpreted as the maximum string length.

The read value is interpreted as a pointer to a string. The character string must end with '\0'. It is output in the dynamic display field.

**Syntax** **CON\_STRING\_LIMIT,**  
**0, 0, par3**

**Note** Strings are truncated and marked with an '\*' only with read access operations.

### 3.5.8 CON\_DECIMAL

**Description** This conversion routine is used to display and input integers and floating-point values.

Depending on the parameter settings, the read value is interpreted as an 8/16/32 bit value, with or without sign, or as a 32-bit floating-point number.

**Syntax** **CON\_DECIMAL,**  
*con\_p1, con\_p2, con\_p3*[*con\_p3*]

**Parameters** *con\_p1* Data format of the value to be displayed:

- F\_BYTE
- F\_UBYTE
- F\_WORD
- F\_UWORD
- F\_LONG
- F\_ULONG
- F\_FLOAT
- F\_DOUBLE

(not with CON\_IO\_RESOLUTION)

*con\_p3* Conversion parameter for decimal values. Individual parameters can be combined.  
(e.g.: CON\_CUT\_ZEROES | CON\_POSITIVE\_SIGN)

CON\_IO\_RESOLUTION  
The number of displayed decimal places is dependent on display machine data \$MM\_DISPLAY\_RESOLUTION.

CON\_NO\_ZEROES  
"0" or "0.0" is shown as an empty field.

CON\_CUT\_ZEROES  
Zeros after the decimal point are cut off.

CON\_POSITIVE\_SIGN  
The value is displayed with a positive sign "+" if it is positive. The default parameterization displays "-", but not "+".

CON\_ABSOLUTE  
Values are not displayed with any sign at all.

The following table shows an overview of meaningful combinations of the various conversion parameters:

**Note**

If the length of the number exceeds the configured width of the input/output field, then you must specify attribute `IO_EDITOR_SCROLL`. You will then be able to scroll in the field using the cursor.

<b>con_p1</b>	<b>con_p2</b>	<b>con_p3</b>
F_DOUBLE F_FLOAT	Number of decimal places (not with CON_IO_RESOLUTION)	CON_IO_RESOLUTION CON_NO_ZEROES CON_CUT_ZEROES CON_POSITIVE_SIGN CON_ABSOLUTE
F_BYTE F_UBYTE F_WORD F_UWORD F_LONG F_ULONG	-	CON_NO_ZEROES CON_CUT_ZEROES CON_POSITIVE_SIGN CON_ABSOLUTE

**Important**

- Only data which are stored in the F\_DOUBLE format may be displayed in this format. The same applies to F\_FLOAT data.
- Notepads to be displayed as a DOUBLE value must be initialized beforehand with a DOUBLE value (routine AC\_SET\_DOUBLE). The same applies to F\_FLOAT data.

### 3.5.9 CON\_HEX

**Description**

This routine is used to display and input values in hexadecimal format.

Depending on the parameter settings, the read value is interpreted as an unsigned 8/16/32-bit value.

**Syntax**

**CON\_HEX,**  
*con\_p1, 0, con\_p3[con\_p3]*

**Parameters**

*con\_p1* Data format of the value to be displayed:  
F\_UBYTE  
F\_UWORD  
F\_ULONG

*con\_p3* Conversion parameter for hexadecimal display. Individual parameters can be combined (e.g.: CON\_DIGIT\_GROUPS | CON\_HEX\_SIGN)

0

No further format conversion.

#### CON\_DIGIT\_GROUPS

Values are displayed in groups of 4 characters (e.g. instead of 020ABC4D -> 020A BC4D).

#### CON\_HEX\_SIGN

A preceding "H" indicates that the value is a hexadecimal number.

#### CON\_NO\_LEADING\_ZEROS

Leading zeros are not inserted in front of the hexadecimal number.

### 3.5.10 CON\_BINARY

<b>Description</b>	<p>This conversion routine is used to display and input values in binary format (bit stream).</p> <p>Depending on the parameter settings, the read value is interpreted as an unsigned 8/16/32-bit value.</p>
<b>Syntax</b>	<p><b>CON_BINARY,</b> <i>con_p1</i>, <b>0</b> , <i>con_p3</i></p>
<b>Parameters</b>	<p><i>con_p1</i> Data format of the value to be displayed: F_UBYTE F_UWORD F_ULONG</p> <p><i>con_p3</i> Conversion parameters for binary display. 0 No further format conversion.</p> <p>CON_DIGIT_GROUPS Values are displayed in groups of 4 characters (e.g. instead of 01011101 -&gt; 0101 1101).</p>

### 3.5.11 CON\_BCD

<b>Description</b>	<p>Input and output of an unsigned 32-bit value as a BCD number. The conversion parameters have no meaning.</p>
<b>Syntax</b>	<p><b>CON_BCD,</b> <b>0, 0, 0</b></p>



### 3.5.12 CON\_BIT

**Description** This conversion routine displays "0" or "1" in the dynamic display field. In this case, the display depends on whether the specified bit is set ("1") or not set ("0") in the supplied value.

The data in which the bit must be checked can be an 8/16/32-bit value depending on parameterization.

**Syntax** **CON\_BIT,**  
*con\_p1, con\_p2, 0*

**Parameters**

<i>con_p1</i>	Data format of the value to be displayed: F_UBYTE F_BYTE F_UWORD F_WORD F_ULONG F_LONG
<i>con_p2</i>	Bit position for the bit to be filtered out in the value to be checked. (0-7/15/31).

### 3.5.13 CON\_OFF

**Description** No data are converted.

**Syntax** **CON\_OFF,**  
**0, 0, 0**



## Notes

## 4

## 4 Action and Reaction Routines

4.1 Routines that affect lists and objects .....	4-142
4.1.1 NEW_MENU: Opening a new menu .....	4-142
4.1.2 NEW_MENU_NB: Opening a menu, indirect identifier (OP 030) .....	4-142
4.1.3 OPEN_WINDOW: Opening a window .....	4-143
4.1.4 OPEN_WINDOW_NB: Opening a window, indirect identifier .....	4-144
4.1.5 CLOSE_WINDOW: Closing a window .....	4-145
4.1.6 CLOSE_WINDOW_NB: Closing a window .....	4-146
4.1.7 REFRESH_WINDOW: Refreshing a window object list .....	4-147
4.1.8 NEW_SOFTKEY: Activating a new soft key line .....	4-148
4.1.9 NEW_SOFTKEY_ASSIGN: Activating a new soft key bar .....	4-149
4.1.10 OPEN_BRC_LIST: Opening a basis reaction list .....	4-149
4.1.11 CLOSE_BRC_LIST: Closing a basic reaction list .....	4-150
4.1.12 OPEN_EVENT_LIST: Opening an event list .....	4-150
4.1.13 CLOSE_EVENT_LIST: Closing an event list .....	4-151
4.1.14 OPEN_LIMIT_LIST: Opening an input limit value list .....	4-151
4.1.15 CLOSE_LIMIT_LIST: Closing an input limit value list .....	4-153
4.1.16 DRAW_OBJECT: Processing an object list .....	4-153
4.1.17 PROCESS_ACTION_LIST: Processing an action list .....	4-154
4.1.18 DRAW_SOFTKEY: Output of a soft key (not HPU) .....	4-155
4.1.19 ACTIVATE_SK_GRAPHIC: Activating the soft key display .....	4-156
4.1.20 DEACTIVATE_SK_GRAPHIC: De-activating the soft key display .....	4-156
4.1.21 ENABLE_SK_VISUALISATION: Withdrawing the soft key display inhibit (06.04.01) .....	4-156
4.1.22 SET_RECALL, RESET_RECALL: Deleting RECALL symbols .....	4-157
4.1.23 SET_MORE, RESET_MORE: Deleting the MORE symbol display (MMC100/EBF) .....	4-157
4.1.24 SET_MORE, RESET_MORE, SET_RECALL, RESET_RECALL .....	4-158
4.1.25 D-CLOSE: Closing an input field with value transfer .....	4-158
4.1.26 D_ABORT: Closing an input field without accepting a value .....	4-159
4.1.27 D_GOTO_DIAFIELD: Positioning the cursor to the dialog field .....	4-159
4.1.28 D_GOTO_DIAFIELD_NB: Positioning the cursor to the dialog field .....	4-160
4.1.29 COPY_CURRENT_DIA_ID .....	4-160
4.1.30 COPY_DIA_ID: Saving the actual cursor position in the notepad .....	4-161
4.1.31 SET_WIN_ATTR, RESET_WIN_ATTR: Changing the attribute in a window .....	4-161
4.1.32 D_SET_DIAFIELD_ATTR: Linking-in attributes .....	4-163
4.1.33 D_RESET_DIAFIELD_ATTR: Resetting dialog field attributes .....	4-165
4.1.34 ACTIVATE_DIA_REFR: Updating dialog fields .....	4-167
4.1.35 D_ACTIVATE_ACTION: Processing the action list .....	4-167
4.1.36 NB_DECREMENT: Decrementing the contents of a notepad entry .....	4-168
4.1.37 NB_INCREMENT: Incrementing the contents of a notepad .....	4-168
4.1.38 SET_TXT_NB: Entering into a text variable .....	4-169
4.1.39 APPEND_TXT_NB_TXT: Attaching text to a text variable .....	4-170

## 4.1 Routines that affect lists and objects

4.1.40	APPEND_TXT_NB_TXT_NB: Attaching text variables .....	4-171
4.1.41	CLEAR_TXT_NB: Deleting a text variable .....	4-171
4.1.42	SCROLL_BAR_REFRESH: Refreshing the scrollbar.....	4-172
4.1.43	SET_EVENT_REACTION: Activating/deactivating an event .....	4-173
4.1.44	BREAK_EVENT: Canceling an event processing .....	4-173
4.1.45	SET_ICON_POS: Setting the user icon bar .....	4-175
4.2	Copying and calculation routines .....	4-175
4.2.1	SET_BIT, RESET_BIT, TOGGLE_BIT: Bit operations.....	4-175
4.2.2	SET_BYTE, SET_WORD, SET_LONG, SET_DOUBLE: Setting a value .....	4-176
4.2.3	CALC_UWORD, CALC_LONG, CALC_DOUBLE: Calculating values ..	4-177
4.2.4	CALC_DATA: Calculating with two variables .....	4-178
4.2.5	COPY_DATA: Copying a variable .....	4-179
4.2.6	COPY_DATA_TO_NB: Copying a variable into the notepad .....	4-179
4.2.7	COPY_BLOCK_NCK_NB: Copying variables blockwise (OP 030).....	4-180
4.2.8	CONVERT_DATA_FORMAT: Converting a data format .....	4-181
4.3	General routines.....	4-182
4.3.1	CHANGE_MODE: NC operating mode change) only OP 030) .....	4-182
4.3.2	CHANNEL_SWITCH: Changing the NC channel .....	4-182
4.3.3	BV_LANGUAGE_CHANGE: Toggling between languages (OP 030)...	4-183
4.3.4	SET_MESSAGE, RESET_MESSAGE: Setting, resetting a message ...	4-183
4.3.5	SET_MSG_POS: Setting the position of the message line (MMC100/EBF).....	4-184
4.3.6	SET_CUR_TXT_POS: Setting the position of the cursor text.....	4-185
4.3.7	SET_INFO, RESET_INFO: Output, delete help symbol .....	4-185
4.3.8	TOOL_SEARCH: Search for tool in the actual TO area (OP 030) .....	4-186
4.3.9	TOOL_CREATE: Create new tool (OP 030) .....	4-186
4.3.10	TOOL_DELETE: Deleting a tool (OP 030) .....	4-187
4.4	Routines to handle part programs.....	4-188
4.4.1	PP_EDIT_OPEN: Opening an edit field for part programs.....	4-188
4.4.2	PP_EDIT_CLOSE: Closing an edit field for part programs .....	4-189
4.4.3	PP_REFRESH: Refreshing an edit field for part programs .....	4-189
4.5	Routines for diagnostics functionality.....	4-190
4.5.1	DG_INIT_ALARM: Initializing for the alarm overview (OP 030) .....	4-190
4.5.2	DG_INIT_MSG: Initializing for the message overview .....	4-191
4.5.3	OPEN_VERSION: Initializing for the NCK version display.....	4-192
4.5.4	CLOSE_VERSION: Closing the NCK version display.....	4-193
4.5.5	DG_INIT_PASSW: Initializing the password dialog.....	4-193
4.5.6	DG_CLOSE_PASSW: Closing the password dialog .....	4-194
4.5.7	DG_SET_PASSW: Setting the password (OP 030).....	4-194
4.5.8	DG_CHG_PASSW: Changing the password for the actual access stage .....	4-195

A range of action and reaction routines are available for implementing functions. They can be activated through configuring measures, i.e. you need to enter the relevant action or reaction elements in the appropriate configuring lists.

---

**Note**

- You may configure actions (AC\_...) only in action lists (ACTION\_LIST), menu/window open list (OPEN\_LIST), menu/window close lists (CLOSE\_LIST) and in the system Init list (SYSTEM\_INIT\_LIST).
  - Reactions (RC\_...) may be configured only in reaction lists (REACTION\_LIST) or soft key reaction lists (SOFTKEY\_REACTION\_LIST).
- 

**AC\_funktion** (*ac\_id*[, *par\_1*, ..., *par\_n*])      /\* Action element \*/

**RC\_funktion** (*rc\_id*, *ev\_code*[, *par\_1*, ..., *par\_n*])      /\* Reaction element \*/

Key:

*Function*                      Function name / type of action or reaction element

*ac\_id / rc\_id*                  Identifier of an action or reaction element

---


**Important**

The identifier must be a constant number. Calculating operations are not permitted at this point.

---

*ev\_code*                      Code of event which must trigger processing of the reaction element.

For a definition of event codes, see:

**References:**      /FB0/, EU, Development Kit  
/PJE/, HMI Embedded Configuring Package

*par\_1 ... par\_n*              Parameters which must be passed to the action or reaction routine. Further details of the number, meaning and possible settings of parameters can be found in this chapter.

The routines are grouped according to function below into

1. routines which affect lists and objects,
2. copy and computation routines,
3. general routines and (e.g. channel switchover)
4. special routines for part program handling.

A description of the execution of each routine as both an action element and a reaction element is given below.

## 4.1 Routines that affect lists and objects

### 4.1.1 NEW\_MENU: Opening a new menu

<b>Description</b>	This routine activates a new menu. First the currently active menu plus all its associated windows is closed in a defined manner through execution of the configured close list(s). When the new menu is activated, the window configured as the first window in the menu is opened first. If configured, the open list to which the menu OPEN_LIST_PTR points is then executed.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_NEW_MENU</b> ( <i>ac_id, menu_id, menu_typ</i> )  <u>Reaction element:</u> <b>RC_NEW_MENU</b> ( <i>rc_id, ev_code, menu_id, menu_typ</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>menu_id</i>	Identifier of the menu definition block to be activated
	<i>menu_typ</i>	Menu type: GLOBAL Global menu. If a new global menu is opened, the previously active local menu is deactivated. LOCAL New local menu. This does not affect the active dialogs of the global menu.
<b>Note</b>	All the attributes defined in the menu definition block and relevant for opening the menu are taken into account.	

### 4.1.2 NEW\_MENU\_NB: Opening a menu, indirect identifier (OP 030)

<b>Description</b>	This routine activates a new menu with an identifier which is not specified directly, but via a notepad entry. The notepad must be set to a defined value beforehand within the configuration (e.g. via an action or a reaction routine).  Before the new menu is opened, the currently active menu plus all its associated windows is closed beforehand in a defined manner through execution of the configured close list(s). When the new menu is activated, the window configured as the first window in the menu is opened first. If configured, the open list to which the menu OPEN_LIST_PTR points is then executed.
--------------------	--

<b>Syntax</b>	<u>Action element:</u> <b>AC_NEW_MENU_NB</b> ( <i>ac_id, nb_nr, menu_typ</i> )	
	<u>Reaction element:</u> <b>RC_NEW_MENU_NB</b> ( <i>rc_id, ev_code, nb_nr, menu_typ</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>nb_nr</i>	Number of notepad which must contain the identifier of the menu definition block to be activated.
	<i>menu_typ</i>	Menu type: GLOBAL  Global menu. If a new global menu is opened, the previously active local menu is deactivated.  LOCAL  New local menu. This does not affect the active dialogs of the global menu.
<b>Note</b>	All the attributes defined in the menu definition block and relevant for opening the menu are taken into account.	

### 4.1.3 OPEN\_WINDOW: Opening a window

<b>Description</b>	This routine opens a window within the global or local menu. If a pointer to an open list is defined within the window definition block, the actions in the list are executed. The objects belonging to the window from the object list and - if configured - from the soft key object list are also displayed on the screen and any specified reaction or soft key reaction lists are activated (see also window definition).	
<b>Syntax</b>	<u>Action element:</u> <b>AC_OPEN_WINDOW</b> ( <i>ac_id, win_id, menu_typ</i> )	
	<u>Reaction element:</u> <b>RC_OPEN_WINDOW</b> ( <i>rc_id, ev_code, win_id, menu_typ</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>win_id</i>	Identifier of the window to be opened.
	<i>menu_typ</i>	Menu type to which window must be assigned.

#### GLOBAL

All windows and object positions are calculated relative to the global menu.

#### LOCAL

All windows and object positions are calculated relative to the local menu.

#### Note

All the attributes defined in the window definition block and relevant for opening the window are taken into account.

The routine operates in the following way:

1. Process the open list (OPEN\_LIST\_PTR of window definition block)
2. Process the object list (OBJECT\_LIST\_PTR of window definition block). All graphics are displayed on the screen in this step.
3. Activate the reaction list (REACTION\_LIST\_PTR of window definition block)
4. Activate the soft key line (SOFTKEY\_OBJECT\_LIST\_PTR and SOFTKEY\_REACTION\_LIST\_PTR of window definition block)

Steps 1 and 2 (process open list and process object list) can be exchanged via the attribute bit **W\_OPEN\_AFTER\_OBJ** in the window definition block. This will be necessary if action routines are called in the open list that access dynamic elements (e.g. IO\_FIELD) which are included in the object list. Examples of such routines are OPEN\_LIMIT\_LIST or PP\_EDIT\_OPEN.



---

#### Important

If further routines are called in the action, reaction or soft key reaction list in which OPEN\_WINDOW is called, these will be executed before the new reaction or soft key reaction list becomes active.

---

### 4.1.4 OPEN\_WINDOW\_NB: Opening a window, indirect identifier

#### Description

This routine opens a window within the global or local menu. The identifier of the window to be opened is not specified directly, but indirectly via a notepad entry. The notepad must be set to a defined value (identifier of the window to be opened) beforehand within the configuration (e.g. via an action or a reaction routine).

The routine behaves in the same way as OPEN\_WINDOW except for the fact that the identifier of the window to be opened is not transferred directly, but within a notepad.



<b>Syntax</b>	<u>Action element:</u> <b>AC_OPEN_WINDOW_NB</b> ( <i>ac_id, nb_nr, menu_typ</i> )  <u>Reaction element:</u> <b>RC_OPEN_WINDOW_NB</b> ( <i>rc_id, ev_code, nb_nr, menu_typ</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>nb_nr</i>	Number of notepad which must contain the identifier of the window to be opened.
	<i>menu_typ</i>	Menu type to which window must be assigned.
		GLOBAL All windows and object positions are calculated relative to the global menu.
		LOCAL All windows and object positions are calculated relative to the local menu.
<b>Note</b>	All the attributes defined in the window definition block and relevant for opening the window are taken into account. See also remarks relating to the OPEN_WINDOW routine	

#### 4.1.5 CLOSE\_WINDOW: Closing a window

<b>Description</b>	This routine closes the specified window in the configured menu. If a pointer to a close list is configured in the window definition block, this list is executed beforehand.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_CLOSE_WINDOW</b> ( <i>ac_id, win_id, menu_typ, cl_mode</i> )  <u>Reaction element:</u> <b>RC_CLOSE_WINDOW</b> ( <i>rc_id, ev_code, win_id, menu_typ, cl_mode</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>win_id</i>	Identifier of the window to be closed.
		0 The active window is closed.
	<i>menu_typ</i>	Menu type from which the window must be removed.
		GLOBAL The window is removed from the global menu.

	LOCAL	
		The window is removed from the local menu.
<i>cl_mode</i>	Mode for closing the window	
	0	
		The window graphic is not deleted.
	WIN_CLOSE_DEL_BC	
		The window area is filled with the background color from the window.
	WIN_CLOSE_KEEP_SK	
		The soft keys of the closed window are retained. This function is meaningful only if a new soft key menu is assigned immediately afterwards. Its purpose is runtime optimization and it prevents unnecessary flickering (used by ShopMill).

**Note** All the attributes defined in the window definition block and relevant for closing the window are taken into account.

#### 4.1.6 CLOSE\_WINDOW\_NB: Closing a window

**Description** This routine closes the window specified in a notepad in the configured menu. If a pointer to a close list is configured in the window definition block, this list is executed beforehand.

**Syntax**

Action element:  
**AC\_CLOSE\_WINDOW\_NB** (*ac\_id, nb, menu\_typ, cl\_mode*)

Reaction element:  
**RC\_CLOSE\_WINDOW\_NB** (*rc\_id, ev\_code, nb, menu\_typ, cl\_mode*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>nb</i>	Number of notepad which must contain the identifier of the window to be closed.
<i>menu_typ</i>	Menu type from which the window must be removed.
	GLOBAL
	The window is removed from the global menu.
	LOCAL
	The window is removed from the local menu.

<i>cl_mode</i>	Mode for closing the window
0	The window graphic is not deleted.
1	The window area is filled with the background color from the window.

**Note** All the attributes defined in the window definition block and relevant for closing the window are taken into account.

#### 4.1.7 REFRESH\_WINDOW: Refreshing a window object list

**Description** This function refreshes the specified window. All static and dynamic display elements are displayed again. If a pointer to an open list is also configured for the window, this list can be processed if desired.

**Syntax**

Action element:  
**AC\_REFRESH\_WINDOW** (*ac\_id, win\_id, menu\_typ, opl\_mode*)

Reaction element:  
**RC\_REFRESH\_WINDOW** (*rc\_id, ev\_code, win\_id, menu\_typ, opl\_mode*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>win_id</i>	Identifier of window to be refreshed. It must be the identified of an open window.
0	The current window is refreshed.
<i>menu_typ</i>	Menu type for which window is opened.
GLOBAL	The window is included in the global menu.
LOCAL	The window is included in the local menu.
<i>opl_code</i>	Identifier which defines whether an open list configured for the window must be processed when the window is refreshed.
NO_OPEN_LIST	No open list is processed when the window is refreshed.

#### DO\_OPEN\_LIST

Any open list configured for the window will be processed when it is refreshed. Whether this list must be activated before or after the display objects are output is taken into account (as specified in the *attr* parameter of the window definition block).

#### ONLY\_FIELD\_REFRESH

No open list or object list is processed when the window is refreshed. The input/output fields are retained and updated.

**Note**                    **The function refreshes only the object list of the window referenced in the window definition block and the object lists it includes. Any elements output temporarily with the DRAW\_OBJECT function are not displayed again.**

### 4.1.8      **NEW\_SOFTKEY: Activating a new soft key line**

**Description**                    This routine activates a new soft key line including the associated soft key reaction list.

**Syntax**                         Action element:  
**AC\_NEW\_SOFTKEY** (*ac\_id, sk\_obl\_id, sk\_rcl\_id*)  
Reaction element:  
**RC\_NEW\_SOFTKEY** (*rc\_id, ev\_code, sk\_obl\_id, sk\_rcl\_id*)

**Parameters**                    *ac\_id, rc\_id*                    Unique identifier of the action or reaction element.  
*ev\_code*                        Code of event which must trigger processing of the reaction element.  
*sk\_obl\_id*                       Unique identifier of soft key object list to be activated.  
*sk\_rcl\_id*                       Unique identifier of soft key reaction list to be deactivated.

**Note**                              The user list directory (file *ap\_l\_dir.h*) must contain the following pointer to both configured lists.  
EXTERN\_SOFTKEY\_OBJECT\_LIST (*sk\_obl\_id*)  
EXTERN\_SOFTKEY\_REACTION\_LIST (*sk\_rcl\_id*)

### 4.1.9 NEW\_SOFTKEY\_ASSIGN: Activating a new soft key bar

**Description** This routine assigns a soft key menu to a window located in the chain of the local menu. This is not activated until the focus is placed on the window. If the focus is currently on the relevant window, the routine behaves like NEW\_SOFTKEY.  
The specified IDs of the soft key object list and soft key reaction list must be included in the list directory.

**Syntax**

Action element:  
**AC\_NEW\_SOFTKEY\_ASSIGN** (*id, sko, skr, win*)

Reaction element:  
**AC\_NEW\_SOFTKEY\_ASSIGN** (*id, event, sko, skr, win*)

**Parameters**

<i>id</i>	Unique identifier of the element within the module.
<i>event</i>	Event where the routine should be triggered.
<i>sko</i>	ID of the soft key object list (must be included in the list directory).
<i>skr</i>	ID of soft key reaction list (must be included in the list directory).
<i>win</i>	ID of window to which soft key menu must be assigned.

**Example** **AC\_NEW\_SOFTKEY\_ASSIGN** (200, MY\_SKO, MY\_SKR, MY\_WIN)

### 4.1.10 OPEN\_BRC\_LIST: Opening a basis reaction list

**Description** Once this routine has been called, a reaction list is registered with the system as a dialog-independent basic reaction list. It thus remains active until it is closed again by the CLOSE\_BRC\_LIST routine.

**Syntax**

Action element:  
**AC\_OPEN\_BRC\_LIST** (*ac\_id, rcl\_id*)

Reaction element:  
**RC\_OPEN\_BRC\_LIST** (*rc\_id, ev\_code, rcl\_id*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>rcl_id</i>	Unique identifier of the reaction list to be opened.

**Note** A maximum of 5 basic reaction lists can be active at the same time.  
The user list directory (file *ap\_l\_dir.h*) must contain the following pointer to the configured reaction list: EXTERN\_REACTION\_LIST (*rcl\_id*).

#### 4.1.11 CLOSE\_BRC\_LIST: Closing a basic reaction list

**Description** An active basic reaction list is closed again when this routine is called.

**Syntax** Action element:  
**AC\_CLOSE\_BRC\_LIST** (*ac\_id, rcl\_id*)  
Reaction element:  
**RC\_CLOSE\_BRC\_LIST** (*rc\_id, ev\_code, rcl\_id*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>rcl_id</i>	Unique identifier of the reaction list to be closed.

#### 4.1.12 OPEN\_EVENT\_LIST: Opening an event list

**Description** This function registers a dialog-independent result list with the system. After the list is opened, internal events are generated according to the event elements configured in the list. These can, in turn, activate reaction elements. An event list remains active until it is closed again by the CLOSE\_EVENT\_LIST routine.

**Syntax** Action element:  
**AC\_OPEN\_EVENT\_LIST** (*ac\_id, evl\_id*)  
Reaction element:  
**RC\_OPEN\_EVENT\_LIST** (*rc\_id, ev\_code, evl\_id*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>evl_id</i>	Unique identifier of the event list to be opened.

**Note** A maximum of 5 event lists can be active at the same time.  
The user list directory must contain the following pointer to the configured event list: EXTERN\_EVENT\_LIST (*evl\_id*)

### 4.1.13 CLOSE\_EVENT\_LIST: Closing an event list

<b>Description</b>	An active event list is closed again when this routine is called.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_CLOSE_EVENT_LIST</b> ( <i>ac_id, evl_id</i> )  <u>Reaction element:</u> <b>RC_CLOSE_EVENT_LIST</b> ( <i>rc_id, ev_code, evl_id</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>evl_id</i>	Unique identifier of the event list to be closed.

### 4.1.14 OPEN\_LIMIT\_LIST: Opening an input limit value list

<b>Description</b>	This routine allocates an input limit-value list to a window. Only one input limit-value list can be assigned to each window at any given time. The input limit-value list is automatically deactivated when the window is closed, but it can also be deactivated at any other time by the CLOSE_LIMIT_LIST routine.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_OPEN_LIMIT_LIST</b> ( <i>ac_id, ll_id, menu_typ, win_id</i> )  <u>Reaction element:</u> <b>RC_OPEN_LIMIT_LIST</b> ( <i>rc_id, ev_code, ll_id, menu_typ, win_id</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>ll_id</i>	Identifier of the input limit value list to be opened.
	<i>menu_typ</i>	Menu type GLOBAL The window to which the limit-value list must be assigned is located in the global menu. LOCAL The window to which the limit-value list must be assigned is located in the local menu.

## 4.1 Routines that affect lists and objects

*win\_id* Identifier of a **window that has already been opened** that should be assigned the input limit value list.

**Note**

The window to which the input limit-value list must be assigned must be opened in the specified menu and contain dialog fields. The window definition block must contain attribute **W\_OPEN\_AFTER\_OBJ**.

The user list directory (file *ap\_l\_dir.h*) must contain the following pointer to the configured input limit-value list:

EXTERN\_LIMIT\_LIST (*ll\_id*)

**Example**

```

BEGIN_OBJECT_LIST (OB_INOUT)
RECTANGLE (102, 0, 0, WIDTH_APLWIN, HEIGHT_APLWIN,
           FILLED, BLACK, 0xff)
TEXT (101, X_T_INOUT, Y_T_INOUT, T_PJ_INOUT,
      CS_SMALL, 0, WHITE)
IO_FIELD (102, X_T_INOUT, Y_T_INOUT+10,8, /* id,x,y,w */
          WHITE, BLACK, /* colors */
          CS_SMALL,
          0, /* field_attr */
          103, 103, 103, 103, /* cur_r, cur_l, cur_d, cur_u,*/
          0, /* acc_class,*/
          10, /* Refr cycle */
          T_PJ_FELD1, /* cursor_txt_id, */
          P_NB, NB_TEST_INP, 0, 0, /*v_adr */
          CON_DECIMAL, F_LONG, 0, 0) /*con-func */
END_OBJECT_LIST (OB_INOUT)
BEGIN_LIMIT_LIST (LIMIT_LIST_INOUT)
LIMIT_ELEMENT (120, 102, LIMIT_CHECK_ON, 100.0, 10000.0)
END_LIMIT_LIST (LIMIT_LIST_INOUT)

BEGIN_OPEN_LIST (OP_W_INOUT)
AC_OPEN_LIMIT_LIST (303, LIMIT_LIST_INOUT, LOCAL,
                    W_INOUT)
END_OPEN_LIST (OP_W_INOUT )
BEGIN_WINDOW (W_INOUT)
W_OPEN_AFTER_OBJ, /*attribute */
X_W_INOUT, Y_W_INOUT, /* x/y positlINOUT */
WIDTH_APLWIN, HEIGHT_APLWIN, /* width, height */
WHITE, /* border color */
BLACK, /* background color */
NULL, /* channel-group */
OPEN_LIST_PTR (OP_W_INOUT), /* pointer to open list */
NULL, /* pointer to Close-List */
OBJECT_LIST_PTR (OB_INOUT), /* pointer to object list */

```



```

NULL,          /* pointer to Reaction-List */
NULL,          /*pointer to SKO-List */
NULL,          /* pointer to SKR-List */
END_WINDOW( W_INOUT )

```

#### 4.1.15 CLOSE\_LIMIT\_LIST: Closing an input limit value list

<b>Description</b>	This routine closes the input limit-value list assigned to a window.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_CLOSE_LIMIT_LIST</b> ( <i>ac_id, menu_typ, win_id</i> ) <u>Reaction element:</u> <b>RC_CLOSE_LIMIT_LIST</b> ( <i>rc_id, ev_code, menu_typ, win_id</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>menu_typ</i>	Menu type GLOBAL The window to which the limit-value list is assigned must be sought in the global menu. LOCAL The window to which the limit-value list is assigned must be sought in the local menu.
	<i>win_id</i>	Identifier of window to which the input limit-value list to be closed is assigned.

#### 4.1.16 DRAW\_OBJECT: Processing an object list

<b>Description</b>	This routine displays the graphic objects of a single object list on the screen. The list must not contain any dynamic elements.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_DRAW_OBJECT</b> ( <i>ac_id, obl_id, menu_typ, win_id</i> ) <u>Reaction element:</u> <b>RC_DRAW_OBJECT</b> ( <i>rc_id, ev_code, obl_id, menu_typ, win_id</i> )	

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>obl_id</i>	Identifier of the object list to be processed.
	<i>menu_typ</i>	Menu type
		GLOBAL The window in which the graphic objects must be displayed is located in the global menu.
	LOCAL The window in which the graphic objects must be displayed is located in the local menu.	
	<i>win_id</i>	Identifier of an open window in which the graphic objects must be displayed. The positions of graphic objects are calculated relative to the position of this window.

**Note**

The window in which the graphic objects must be displayed must already be open in the specified menu.

Objects displayed in this way must not be placed over display objects located in a window object list, because these are automatically refreshed by the system and the objects from the DRAW\_OBJECT list will thus be overwritten repeatedly.

The user list directory (file *ap\_l\_dir.h*) must contain the following pointer to the configured object list: EXTERN\_OBJECT\_LIST (*obl\_id*).

Unlike the objects in a window object list, objects displayed with DRAW\_OBJECT are not automatically refreshed by the system.

#### 4.1.17 PROCESS\_ACTION\_LIST: Processing an action list

**Description** This routine processes an action list. It is used when a series of identical actions need to be performed in several reaction, open or close lists.

**Syntax**

Action element:  
**AC\_PROCESS\_ACTION\_LIST** (*ac\_id, acl\_id*)

Reaction element:  
**RC\_PROCESS\_ACTION\_LIST** (*rc\_id, ev\_code, acl\_id*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>acl_id</i>	Identifier of the action list to be processed.

**Note** The action list to be processed must be entered in the user list directory (file *ap\_l\_dir.h*).  
EXTERN\_ACTION\_LIST (*acl\_id*)

#### 4.1.18 DRAW\_SOFTKEY: Output of a soft key (not HPU)

**Description** This routine displays a soft key on the screen. This is always a horizontal soft key on OP 030s, but can be applied to both horizontal and vertical soft keys on larger operator panels (e.g. MMC 100/UOP).

**Syntax** Action element:  
**AC\_DRAW\_SOFTKEY** (*ac\_id*, *sk\_txt\_nr*, *sk*, *sk\_lines*, *sk\_sts*)  
Reaction element:  
**RC\_DRAW\_SOFTKEY** (*rc\_id*, *ev\_code*, *sk\_txt\_nr*, *sk*, *sk\_lines*, *sk\_sts*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>sk_txt_nr</i>	Number of text to be displayed from a text list.
<i>sk</i>	Identifier / designation of the soft key to be displayed. KEY_F1 ... KEY_F8       =>horizontal soft key KEY_F1_V ... KEY_F8_V =>vertical soft key
<i>sk_lines</i>	Number of text lines of the soft key.
<i>sk_sts</i>	Status of the soft key. PRESSED, SK_PRESSED The soft key is displayed in reverse video. NOT_PRESSED, SK_NOT_PRESSED The soft key is displayed normally. SK_DEACTIVATED The soft key is displayed with semi-concealed print on the screen. It is deactivated.

**Note** Soft key displays are **not** supported by a system function on the handheld programming unit (HPU), see Chapter 3.

#### 4.1.19 ACTIVATE\_SK\_GRAPHIC: Activating the soft key display

**Description** This routine reactivates the soft key display and shows the current SK menu on the screen.  
Afterwards, the soft key graphic is output immediately for all new windows and soft key menus.

**Syntax**

Action element:  
AC\_ACTIVATE\_SK\_GRAPHIC (*ac\_id*)

Reaction element:  
RC\_ACTIVATE\_SK\_GRAPHIC (*rc\_id, event*)

**Parameters**

*ac\_id, rc\_id* Unique identifier of the action or reaction element.  
*event* Event for which the routine should be initiated.

#### 4.1.20 DEACTIVATE\_SK\_GRAPHIC: De-activating the soft key display

**Description** This routine deactivates the soft key display. No soft keys are output subsequently until the soft key display is enabled again by the ACTIVATE\_SK\_GRAPHIC routine. Deactivating the soft key display is useful for preventing a SK menu from being displayed for each individual window when several windows are opened at once.  
The disable command is not canceled again until the last SK menu has been output (ACTIVATE\_SK\_GRAPHIC), in which case the SK menu of the active window is displayed.

Caution: Soft keys configured with DRAW\_SOFTKEY are NOT displayed after reactivation.

**Syntax**

Action element:  
AC\_DEACTIVATE\_SK\_GRAPHIC (*ac\_id*)

Reaction element:  
RC\_DEACTIVATE\_SK\_GRAPHIC (*rc\_id, event*)

**Parameters**

*ac\_id, rc\_id* Unique identifier of the action or reaction element.  
*event* Event for which the routine should be initiated.

#### 4.1.21 ENABLE\_SK\_VISUALISATION: Withdrawing the soft key display inhibit (06.04.01)

**Description** This routine withdraws the soft key display inhibit for subsequently requested soft keys. Softkeys that were requested while the soft key display was inhibited, are nicht updated on the screen (contrary to ACTIVATE\_SK\_GRAPHIC).

**Syntax**

Action element:  
AC\_ENABLE\_SK\_VISUALISATION (*id*)

Reaction element:  
RC\_ENABLE\_SK\_VISUALISATION (*id, event*)

**Parameters**

*id* Unique identifier of the action or reaction element.

*event* Event for which the routine should be initiated.

#### 4.1.22 SET\_RECALL, RESET\_RECALL: Deleting RECALL symbols

**Description**

These routines show or delete the Recall symbol. The display positions are predefined.

The Recall symbol (^) indicates whether you are at a level other than the main operating level and thus whether you can switch back to the main operating level.

**Syntax**

Action element:  
AC\_SET\_RECALL (*ac\_id*)  
AC\_RESET\_RECALL (*ac\_id*)

Reaction element:  
RC\_SET\_RECALL (*rc\_id, ev\_code*)  
RC\_RESET\_RECALL (*rc\_id, ev\_code*)

**Parameters**

*ac\_id, rc\_id* Unique identifier of the action or reaction element.

*ev\_code* Code of event which must trigger processing of the reaction element.

#### 4.1.23 SET\_MORE, RESET\_MORE: Deleting the MORE symbol display (MMC100/EBF)

**Description**

These routines show or delete the More symbol. The display positions are predefined.

The More symbol (>) indicates whether you can switch to other areas (more than 8) (area switchover).

**Syntax**

Action element:  
AC\_SET\_MORE (*ac\_id*)  
AC\_RESET\_MORE (*ac\_id*)

Reaction element:  
RC\_SET\_MORE (*rc\_id, ev\_code*)  
RC\_RESET\_MORE (*rc\_id, ev\_code*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.

#### 4.1.24 SET\_MORE, RESET\_MORE, SET\_RECALL, RESET\_RECALL

**Description** This routine sets or activates the Recall/More symbol from the object/soft key object list. You can thus assign the graphic for the symbols directly to the graphic for the soft keys. For more information, see also Subsections 4.1.21 and 4.1.22.

**Syntax**

Action element:  
AC\_SET\_RECALL (*ac\_id*)  
AC\_RESET\_RECALL (*ac\_id*)  
AC\_SET\_MORE (*ac\_id*)  
AC\_RESET\_MORE (*ac\_id*)

Reaction element:  
RC\_SET\_RECALL (*rc\_id, event*)  
RC\_RESET\_RECALL (*rc\_id, event*)  
RC\_SET\_MORE (*rc\_id, event*)  
RC\_RESET\_MORE (*rc\_id, event*)

Within object/soft key object lists (new)  
OB\_C\_W\_SKIP\_IF (*ID, 0, 0 P\_SET\_RECALL, 0, 0, 0*)  
OB\_C\_W\_SKIP\_IF (*ID, 0, 0 P\_RESET\_RECALL, 0, 0, 0*)  
OB\_C\_W\_SKIP\_IF (*ID, 0, 0 P\_SET\_MORE, 0, 0, 0*)  
OB\_C\_W\_SKIP\_IF (*ID, 0, 0 P\_RESET\_MORE, 0, 0, 0*)

**Parameters** *ac\_id, rc\_id* Unique identifier of the action or reaction element.



---

**Important**

The characters in bold print must be entered unchanged; they are constants rather than variables. The skip function is used only to reach the Recall/More function, but they are not skipped.

---

#### 4.1.25 D-CLOSE: Closing an input field with value transfer

**Description** This routine closes an input field. On this changeover from edit to dialog mode, the value stored in the input field is written to the configured data destination, the edit cursor is deleted and the dialog cursor activated again.

The value is accepted without actuation of the Input, Return or cursor keys, for example, in conjunction with an "OK" soft key.

<b>Syntax</b>	<u>Action element:</u> <b>AC_D_CLOSE</b> ( <i>ac_id</i> )	
	<u>Reaction element:</u> <b>RC_D_CLOSE</b> ( <i>rc_id, ev_code</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<b>Note</b>	The entered value may be lost when you exit the screen or switch to another area. It is therefore advisable to specify the D_CLOSE command in the close list of a window.	

#### 4.1.26 D\_ABORT: Closing an input field without accepting a value

<b>Description</b>	This routine switches from edit to dialog mode. On switchover, any inputs you have made are discarded, the edit cursor is deleted and the dialog cursor activated.	
	The value is not transferred, for example, in conjunction with a "Cancel" soft key".	
<b>Syntax</b>	<u>Action element:</u> <b>AC_D_ABORT</b> ( <i>ac_id</i> )	
	<u>Reaction element:</u> <b>RC_D_ABORT</b> ( <i>rc_id, ev_code</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.

#### 4.1.27 D\_GOTO\_DIAFIELD: Positioning the cursor to the dialog field

<b>Description</b>	This routine places the dialog cursor within the current window on the configured dialog field. If '0' or the number of a field which does not exist in the window is transferred as the dialog field number, the dialog cursor is positioned on the first configured dialog field in the window. If 0 is passed as the field number and the W_STORE_CURSOR_LOCATION attribute is configured in the window definition block, the dialog cursor returns to the dialog field on which it was originally positioned when the window is selected again
--------------------	--

<b>Syntax</b>	<u>Action element:</u> <b>AC_D_GOTO_DIAFIELD</b> ( <i>ac_id, df_id</i> )	
	<u>Reaction element:</u> <b>RC_D_GOTO_DIAFIELD</b> ( <i>rc_id, ev_code, df_id</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>df_id</i>	Identifier of the dialog box into which the dialog cursor should be set (refer to the Description).

#### 4.1.28 D\_GOTO\_DIAFIELD\_NB: Positioning the cursor to the dialog field

**Description** This routine places the dialog cursor within the current window on the configured dialog field.  
The dialog field number is transferred using a notepad. Otherwise, the function behaves like D\_GOTO\_DIAFIELD.  
The dialog number can, for example, be downloaded into the notepad with the routines COPY\_DIA\_ID or COPY\_CURRENT\_DIA\_ID.

**Syntax** Action element:  
**AC\_D\_GOTO\_DIAFIELD\_NB** (*ac\_id, nb\_df\_id*)  
Reaction element:  
**RC\_D\_GOTO\_DIAFIELD\_NB** (*rc\_id, ev\_code, nb\_df\_id*)

**Parameters** *ac\_id, rc\_id* Unique identifier of the action or reaction element.  
*ev\_code* Code of event which must trigger processing of the reaction element.  
*nb\_df\_id* Notepad with the identifier of the dialog field in which the dialog cursor should be set.

#### 4.1.29 COPY\_CURRENT\_DIA\_ID

**Description** This routine copies the identity of the dialog field in which the cursor is positioned to a notepad **before** the cursor is moved to another field.  
COPY\_DIA\_ID has the same effect, but **after** the cursor has left the field. The two routines therefore differ only in terms of the events which move the cursor, but are identical in relation to other events. When configured as an action element, the distinction applies only if the routine is called in an action list as a result of a cursor event which, in turn, has been called from a reaction list.  
If no cursor exists, the routine returns the value '10' in the specified notepad.

**Operating sequence with navigation events:**



**COPY\_DIA\_ID:**

1. Move cursor – 2. Write dialog field\_ID to notepad

**COPY\_CURRENT\_DIA\_ID:**

1. Write dialog field\_ID to notepad – 2. Move cursor

**Syntax**Action element:**AC\_COPY\_CURRENT\_DIA\_ID** (*id, nb*)Reaction element:**RC\_COPY\_CURRENT\_DIA\_ID** (*id, event, nb*)**Parameters**

<i>id</i>	Unique identifier of the element within the module.
<i>event</i>	Event for which the routine should be initiated.
<i>nb</i>	Number of notepad to which dialog field identity must be written.

**4.1.30 COPY\_DIA\_ID: Saving the actual cursor position in the notepad****Description**

This routine writes the identifier of the dialog field on which the dialog cursor is positioned to a notepad.

**Syntax**Action element:**AC\_COPY\_DIA\_ID** (*ac\_id, nb\_nr*)Reaction element:**RC\_COPY\_DIA\_ID** (*rc\_id, ev\_code, nb\_nr*)**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>nb_nr</i>	Number of notepad entry to which identifier of the current dialog field is written.

**4.1.31 SET\_WIN\_ATTR, RESET\_WIN\_ATTR: Changing the attribute in a window****Description**

This routine changes attributes in a window. It checks whether the relevant attribute bits may be altered from the configuration. This routine has no effect if an error is detected (window not found, invalid menu, invalid attribute).

<b>Syntax</b>	<u>Action element:</u> <b>AC_SET_WIN_ATTR</b> ( <i>id, attr, win, menu</i> )  <u>Reaction element:</u> <b>AC_SET_WIN_ATTR</b> ( <i>id, event, attr, win, menu</i> )
<b>Parameters</b>	<p><i>id</i> Unique identifier of the element within the module.</p> <p><i>event</i> Event for which the routine should be initiated.</p> <p><i>attr</i> Windows attribute (defined in attr.h). The attributes can be OR'ed bitwise. They can be set as follows:</p> <p><b>W_STORE_CURSOR_LOCATION</b> saves the cursor position.</p> <p><b>W_REFRESH_LAST_SOFTKEY</b> causes the soft key to be refreshed before opening the window was active</p> <p><b>W_FOCUS_DISABLE</b> prevents the window from becoming the active focus.</p> <p><b>W_LOCK_FOCUS</b> locks the focus on this window.</p> <p><b>W_INTELLI_CURSOR</b> activates the intelligent cursor control.</p> <p><i>win</i> ID of the window which is to be kept active.</p> <p><i>menu</i> Menu type, possible values: <b>LOCAL</b>, <b>GLOBAL</b>.</p>
<b>Example</b>	<b>AC_SET_WIN_ATTR</b> (200, W_STORE_CURSOR_LOCATION / W_LOCK_FOCUS, 342000, LOCAL)
<b>Intelligent cursor control</b>	<p>If the intelligent cursor control is activated statically with W_INTELLI_CURSOR in the window configuration, or dynamically with AC_/RC_SET_WIN_ATTR, the cursor navigation parameters within the dialog fields are no longer taken into account.</p> <p>The function can be activated and deactivated dynamically while the window is active. When the intelligent cursor control is active, the cursor behaves as follows:</p> <p><b>Cursor down:</b> The cursor is positioned on the nearest cursor field below the current field which is superimposed on the current field in a vertical direction.</p> <p><b>Cursor up:</b> The cursor is positioned on the nearest cursor field above the current field which is superimposed on the current field in a vertical direction.</p>

**Cursor right:**

The cursor is positioned on the nearest cursor field to the right of the current field which is superimposed on the current field in a horizontal direction.

**Cursor left:**

The cursor is positioned on the nearest cursor field to the left of the current field which is superimposed on the current field in a horizontal direction.

**Cursor end:**

The cursor is positioned on the cursor field which the greatest y position. If there are several fields in this position, the cursor is placed on the field furthest to the right.

**Cursor home:**

The cursor is positioned on the cursor field which the lowest y position. If there are several fields in this position, the cursor is placed on the field furthest to the left.

The following applies to all fields: The cursor only jumps to them if they are not inhibited. The cursor remains at its present position if, in the selected direction, there is no field that manifests coverage with the actual field.

**4.1.32 D\_SET\_DIAFIELD\_ATTR: Linking-in attributes**

<b>Description</b>	This routine can be used to set the attributes of one or several dialog fields or one or all windows of a menu. These attributes are set in addition to those that are already active.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_D_SET_DIAFIELD_ATTR(_32)</b> ( <i>ac_id</i> , <i>menu_typ</i> , <i>win_id</i> , <i>df_id_1</i> , <i>df_id_2</i> , <i>df_attr</i>   <i>df_attr...</i> )  <u>Reaction element:</u> <b>RC_D_SET_DIAFIELD_ATTR(_32)</b> ( <i>rc_id</i> , <i>ev_code</i> , <i>menu_typ</i> , <i>win_id</i> , <i>df_id_1</i> , <i>df_id_2</i> , <i>df_attr</i>   <i>df_attr...</i> )	
<b>Parameters</b>	<i>ac_id</i> , <i>rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>menu_typ</i>	Menu type GLOBAL The window or windows in which the dialog field attributes must be modified must be sought in the global menu. LOCAL The window or windows in which the dialog field attributes must be modified must be sought in the local menu.

## 4.1 Routines that affect lists and objects

<i>win_id</i>	Identifier of the window in which the dialog field attributes are to be changed. If 0xffff is entered here, then the attributes of all of the dialog fields are changed in all of the opened windows.
<i>df_id_1</i>	Identifier of the first dialog field in which one or several attributes are to be changed. 0xffff should be entered here if the attribute word of all dialog fields of the window are to be changed.
<i>df_id_2</i>	Identifier of last dialog field in which attribute values must be changed. The number configured here must be >= the identifier of the first field. If 0xffff is specified here, the attributes of all dialog fields of the window are modified irrespective of the identifier configured for the first dialog field ( <i>df_id_1</i> ).
<i>df_attr</i>	Attributes, that should be set in addition to the already existing dialog field attributes. Several attributes can be specified as OR operations ( ). The following dialog field attributes can be modified: IO_CALCULATOR_OFF Deactivate pocket calculator function. IO_EDITOR_SCROLL The editor can scroll the field contents. IO_EDITOR_ALPHA_NUM Alphanumeric values can be input. IO_LEFT_ADJUST The contents are to displayed justified to the left. IO_RIGHT_ADJUST The contents are to be displayed justified to the right. IO_CENTRE_ADJUST The contents are to be displayed centered.

**Note**

If the action routine in the open list of a window is applied to input/output fields defined in the object list of the window, You must specify the attribute **W\_OPEN\_AFTER\_OBJECT** in the window definition block.

For further attributes, please see file ATTR.H.

Please note that different attributes apply to the different types of input and output field. See description of individual input/output fields.

For attributes between the values 0x10000 and 0xFFFFFFFF, you must use macro **SET\_DIA\_FIELD\_ATTR\_32** .




---

**Important**

- By using attributes other than those described above, you intervene significantly in the operation of the system. This may render the system software inoperable.
  - The configuring engineer must thoroughly test any attributes (other than those listed above) before they are used!
- 

### 4.1.33 D\_RESET\_DIAFIELD\_ATTR: Resetting dialog field attributes

**Description** This routine can be used to reset the attributes of one or several dialog fields or one or all windows. Only the attributes you specify are reset, the others remain unaffected.

**Syntax**

Action element:  
**AC\_D\_RESET\_DIAFIELD\_ATTR(\_32)** (*ac\_id, menu\_typ, win\_id, df\_id\_1, df\_id\_2, df\_attr[ | df\_attr.]*)

Reaction element:  
**RC\_D\_RESET\_DIAFIELD\_ATTR(\_32)** (*rc\_id, ev\_code, menu\_typ, win\_id, df\_id\_1, df\_id\_2, df\_attr[ | df\_attr.]*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>menu_typ</i>	Menu type GLOBAL The window or windows in which the dialog field attributes must be reset must be sought in the global menu. LOCAL The window or windows in which the dialog field attributes must be reset must be sought in the local menu.
	<i>win_id</i>	Identifier of the window in which the dialog field attributes are to be changed. If 0xffff is entered here, then the attributes of all of the dialog fields are changed in all of the opened windows.
	<i>df_id_1</i>	Identifier of the first dialog field in which one or several attributes are to be changed. 0xffff should be entered here if the attribute word of all dialog fields of the window are to be changed.
	<i>df_id_2</i>	Identifier of last dialog field in which attribute values must be changed. The number configured here must be >= the identifier of the first field. If 0xffff is specified here, the attributes of all dialog fields of the window are modified irrespective of the identifier configured for the first dialog field ( <i>df_id_1</i> ).

<i>df_attr</i>	Attribute to be reset. Several attributes can be specified as OR operations ( ). The following dialog field attributes can be reset:  IO_CALCULATOR_OFF Deactivate pocket calculator function.  IO_EDITOR_SCROLL The editor can scroll the field contents.  IO_EDITOR_ALPHA_NUM Alphanumeric values can be input.  IO_LEFT_ADJUST The content must be displayed justified to the left.  IO_RIGHT_ADJUST The content must be displayed justified to the right.  IO_CENTRE_ADJUST The content must be displayed centered.
----------------	---

**Note**

If a justification attribute (IO\_RIGHT\_ADJUST, IO\_CENTRE\_ADJUST or IO\_LEFT\_ADJUST) is included in the parameter *df\_attr*, the justification attribute of the fields involved are always set to the default alignment. This means, that for fields, that contain strings (conversion function = CON\_STRING, CON\_STRING\_LIMIT, CON\_ASCII, or CON\_TEXT), then the display is left justified; for fields, that contain numerical values, the display is right justified.

If the action routine in the open list of a window is applied to input/output fields defined in the object list of the window, You must specify the attribute **W\_OPEN\_AFTER\_OBJECT** in the window definition block.

For further attributes, please see file ATTR.H.

Please note that different attributes apply to the different types of input and output field. See description of individual input/output fields.

For attributes between the values 0x10000 and 0xFFFFFFFF, you must use macro **RESET\_DIA\_FIELD\_ATTR\_32**.



---

**Important**

By using attributes other than those described above, you intervene significantly in the operation of the system. This may render the system software inoperable.

The configuring engineer must thoroughly test any attributes (other than those listed above) before they are used!

---

### 4.1.34 ACTIVATE\_DIA\_REFR: Updating dialog fields

<b>Description</b>	This routine refreshes one or several dialog fields in one or all windows of a menu.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_ACTIVATE_DIA_REFR</b> ( <i>ac_id, menu_typ, win_id, df_id</i> ) <u>Reaction element:</u> <b>RC_ACTIVATE_DIA_REFR</b> ( <i>rc_id, ev_code, menu_typ, win_id, df_id</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>menu_typ</i>	Menu type GLOBAL The window or windows containing dialog fields to be refreshed must be sought in the global menu. LOCAL The window or windows containing dialog fields to be refreshed must be sought in the local menu.
	<i>win_id</i>	Identifier of window in which dialog fields must be refreshed. When 0xffff is specified, all dialog fields of all open windows are refreshed.
	<i>df_id</i>	Identifier of dialog field to be refreshed. Specify 0xffff to refresh all dialog fields of the window.

### 4.1.35 D\_ACTIVATE\_ACTION: Processing the action list

<b>Description</b>	This routine starts processing of the action list configured in the action field on which the dialog cursor is currently positioned.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_D_ACTIVATE_ACTION</b> ( <i>ac_id</i> ) <u>Reaction element:</u> <b>RC_D_ACTIVATE_ACTION</b> ( <i>rc_id, ev_code</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.

### 4.1.36 NB\_DECREMENT: Decrementing the contents of a notepad entry

**Description** This routine reduces the value in a notepad entry by the configured, constant value. The value is not reduced to less than the lower limit value specified.

This function supports paging mechanisms in cases where the content of the notepad entry has indexed access to data displayed in dialog fields.

**Syntax**

Action element  
**AC\_NB\_DECREMENT** (*ac-id, nb\_nr, plim\_l, dist*)

Reaction element  
**RC\_NB\_DECREMENT** (*rc\_id, ev\_code, nb\_nr, plim\_l, dist*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>nb_nr</i>	Number of notepad entry whose content must be decremented.
<i>plim_l</i>	Maximum permissible lower limit of value in notepad entry.
<i>dist</i>	Constant value by which the notepad entry content is decremented in each case.



---

**Important**

The notepad must contain a value of data type WORD!

---

### 4.1.37 NB\_INCREMENT: Incrementing the contents of a notepad

**Description** This routine increases the value in a notepad entry by the configured, constant value. The upper limit value passed to the routine is not exceeded.

This function supports paging mechanisms in cases where the content of the notepad entry has indexed access to data displayed in dialog fields.

**Syntax**

Action element:  
**AC\_NB\_INCREMENT** (*ac\_id, nb\_nr, plim\_h, dist*)

Reaction element:  
**RC\_NB\_INCREMENT** (*rc\_id, ev\_code, nb\_nr, plim\_h, dist*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
---------------------	--



<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>nb_nr</i>	Number of notepad entry whose content must be incremented.
<i>plim_h</i>	Maximum permissible upper limit of notepad entry content.
<i>dist</i>	Constant value by which the notepad entry content is incremented in each case.

**Important**

The notepad must contain a value of data type WORD!

---

### 4.1.38 SET\_TXT\_NB: Entering into a text variable

#### Description

This routine finds a text specified by *txt\_nr* and copies it to a text variable defined by *txt\_var\_id*. The text entered in the text variable, including the end identifier, must not exceed 100 characters. Texts exceeding this length will be truncated.

Text variables are structured in the form of a text list with 10 text entries.

**Important**

Two texts are always copied, *txt\_nr* and *txt\_nr+1*, the texts are copied to the text elements *txt\_var\_id* and *txt\_var-id+1*.

The address of the specified text variable is entered in the transferred notepad entry. The address of the next text variable is entered in the following notepad entry.

---

#### Syntax

Action element:

**AC\_SET\_TXT\_NB** (*ac\_id, txt\_nr, nb\_nr, txt\_var\_id*)

Reaction element:

**RC\_SET\_TXT\_NB** (*rc\_id, ev\_code, txt\_nr, nb\_nr, txt\_var\_id*)

Preset one notepad only (**only MMC 100**):

Action element:

**AC\_SET\_TXT\_NB\_SINGLE** (*ac\_id, txt\_nr, nb\_nr, txt\_var\_id*)

Reaction element:

**RC\_SET\_TXT\_NB\_SINGLE** (*rc\_id, ev\_code, txt\_nr, nb\_nr, txt\_var\_id*)

#### Parameters

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>txt_nr</i>	Number of text to be entered in the text variable. The text may be a maximum of 100 characters (including end identifier '\0').

	<i>txt_nr</i> and <i>txt_nr + 1</i> are copied (except for <b>SET_TXT_NB_SINGLE</b> ).
<i>nb_nr</i>	Notepad number and notepad number+1 in which the addresses (e.g. for C function access operations) of the text variables ( <i>txt_var_id</i> and <i>txt_var_id+1</i> ) are returned.  Notepad entries <i>nb_nr</i> and <i>nb_nr + 1</i> are written (except for <b>SET_TXT_NB_SINGLE</b> ).
<i>txt_var_id</i>	Index to a text management list in which text must be entered. 0, 2, 4, 6 and 8 are the only permissible setting values!  <i>txt_var_id</i> and <i>txt_var_id+1</i> are written (except for <b>SET_TXT_NB_SINGLE</b> ).

**Note** The function is used in conjunction with edit fields (EDIT\_FIELD).

### 4.1.39 APPEND\_TXT\_NB\_TXT: Attaching text to a text variable

**Description** This routine finds a text specified by *txt\_nr* and attaches it to a **text variable**. If the total resulting text in the table element exceeds a maximum length of 100 characters, it will be truncated.

The text variable is specified by a notepad entry.



**Important**

The configured notepad entry must be initialized beforehand with the text variable address via an AC/RC\_SET\_TXT\_NB call.

**Syntax**

Action element:  
**AC\_APPEND\_TXT\_NB\_TXT** (*ac\_id, nb\_nr, txt\_nr*)

Reaction element:  
**RC\_APPEND\_TXT\_NB\_TXT** (*rc\_id, ev\_code, nb\_nr, txt\_nr*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>nb_nr</i>	Number of notepad entry containing the address of the text variable to which the text must be appended.
<i>txt_nr</i>	Text number.

**Note** The function is used in conjunction with edit fields.

#### 4.1.40 APPEND\_TXT\_NB\_TXT\_NB: Attaching text variables

**Description** This routine appends the content of one text variable to another text variable. If the total resulting text exceeds a maximum length of 100 characters, it will be truncated.

Each text variable is specified by a notepad entry.

**Important**

The configured notepad entry must be initialized beforehand with the text variable address by an AC/RC \_SET\_TXT\_NB call.

**Syntax**

Action element:

**AC\_APPEND\_TXT\_NB\_TXT\_NB** (*ac\_id, nb\_nr\_z, nb\_nr\_q*)

Reaction element:

**RC\_APPEND\_TXT\_NB\_TXT\_NB** (*rc\_id, ev\_code, nb\_nr\_z, nb\_nr\_q*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>nb_nr_z</i>	Target. Number of notepad entry containing the address of the text variable to which text must be appended.
<i>nb_nr_q</i>	Source. Number of the notepad containing the address of the text variable whose content must be attached to <i>nb_nr_z</i> .

**Note**

The function is used in conjunction with edit fields.

#### 4.1.41 CLEAR\_TXT\_NB: Deleting a text variable

**Description** This routine deletes the content of a text variable. The text variable is specified by a notepad entry.

**Important**

The configured notepad entry must be initialized beforehand with the text variable address via an AC/RC \_SET\_TXT\_NB call.

**Syntax**

Action element:

**AC\_CLEAR\_TXT\_NB** (*ac\_id, nb\_nr*)

Reaction element:

**RC\_CLEAR\_TXT\_NB** (*rc\_id, ev\_code, nb\_nr*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>nb_nr</i>	Number of notepad entry which must contain the address of the text variable to be deleted.

**Note** The function is used in conjunction with edit fields.

#### 4.1.42 SCROLL\_BAR\_REFRESH: Refreshing the scrollbar

**Description** This routine refreshes a scrollbar. The size of the scrollbar may change if the data set to which it refers has altered.

**Syntax**

Action element:  
**AC\_SCROLL\_BAR\_REFRESH** (*ac\_id, sb\_nr, q\_vol, d\_vol, q\_index, d\_index*)

Reaction element:  
**RC\_SCROLL\_BAR\_REFRESH** (*rc\_id, ev\_code, sb\_nr, q\_vol, d\_vol, q\_index, d\_index*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>sb_nr</i>	Number of the scrollbar element that is to be refreshed
	<i>q_vol</i>	Reference to the data source for the complete data quantity NOTEPAD The following parameter <i>d_vol</i> contains a notepad number. The notepad with this number contains the total data set to which the scrollbar refers. Not NOTEBOOK The following parameter <i>d_vol</i> contains the total data set to which the scrollbar refers.
	<i>d_vol</i>	Number of the notepad containing the total data set, or the data set itself (depending on preceding parameter <i>q_vol</i> ).
	<i>q_index</i>	Reference to data source for index to the current data. NOTEPAD The following parameter <i>d_index</i> contains a notepad number. The notepad with this number contains the index to the current data.

Not NOTEBOOK

The following parameter *d\_index* contains the index to the current data.

*d\_index* Number of the notepad containing the index to the current data, or the index to the current data itself (depending on preceding parameter *q\_index*).

#### 4.1.43 SET\_EVENT\_REACTION: Activating/deactivating an event

**Description** This routine activates/deactivates processing of an event within the system basic reaction list. It can be used, for example, to deactivate the default reaction to the machine basic display key and activate it again. All relevant events to which reactions are configured in the system basic reaction list are defined in header file event.h.

The routine is used in the OPEN list of a menu or window.

When you switch to another operating area, the default status is restored automatically, i.e. all events in the system basic reaction list are activated.

**Syntax** Action element:  
**AC\_SET\_EVENT\_REACTION** (*ac\_id*, *event*, *mode*)

The routine can be configured only as an action element.

**Parameters**

<i>ac_id</i>	Unique identifier of the action element
<i>event</i>	Event whose execution must be activated/deactivated in the system basic reaction list
<i>mode</i>	Mode
	EV_NO_REACTION
	Deactivate reaction.
	EV_FULL_REACTION
	Activate reaction.

#### 4.1.44 BREAK\_EVENT: Canceling an event processing

**Description** This routine cancels processing of an event within the system basic reaction list. It is not intended for applications (and thus not for the user), but used exclusively within the system basic reaction list. The system basic reaction list contains general responses to events that are always processed (e.g. actuation of the machine key results in switchover to the MACHINE operating area...). Each application and every window or menu can disable or enable individual

---

4.1 Routines that affect lists and objects

events in the OPEN list for processing in the system basic reaction list (AC\_SET\_EVENT\_REACTION routine). When you switch to another operating area, the default status is restored automatically, i.e. all events in the system basic reaction list are activated.

<b>Syntax</b>	<u>Reaction element</u> <b>RC_BREAK_EVENT</b> ( <i>rc_id</i> , <i>event</i> )	
<b>Parameters</b>	<i>rc_id</i>	Unique identifier of the reaction element.
	<i>event</i>	Event whose processing list is to be cancelled.

#### 4.1.45 SET\_ICON\_POS: Setting the user icon bar

<b>Description</b>	This new configuring function sets the position of the user icon bar.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_SET_ICON_POS</b> ( <i>ac_id</i> , <i>bar_num</i> , <i>x_pos</i> , <i>y_pos</i> ) <u>Reaction element:</u> <b>RC_SET_ICON_POS</b> ( <i>rc_id</i> , <i>event</i> , <i>bar_num</i> , <i>x_pos</i> , <i>y_pos</i> )	
<b>Parameters</b>	<i>ac_id</i> , <i>rc_id</i>	Unique identifier of the action or reaction element.
	<i>bar_num</i>	Specifies the number of the user icon bar, starting with 1 (only 1 bar is supported in the current version). Each bar can accommodate 16 icons.
	<i>x_pos</i> , <i>y_pos</i>	Specifies the top, left-hand corner of the icon bar.

## 4.2 Copying and calculation routines

### 4.2.1 SET\_BIT, RESET\_BIT, TOGGLE\_BIT: Bit operations

<b>Description</b>	These routines can be used to set, reset or invert one or more bits in a data configured via parameters <i>v_adr</i> and <i>v_p1</i> ... <i>v_p3</i> . All bits to be modified are transferred as a bit mask in the relevant call (parameter <i>bit_mask</i> ).	
<b>Syntax</b>	<u>Action elements:</u> <b>AC_SET_BIT</b> ( <i>ac_id</i> , <i>bit_mask</i> , <i>v_adr</i> , <i>v_p1</i> , <i>v_p2</i> , <i>v_p3</i> ) <b>AC_RESET_BIT</b> ( <i>ac_id</i> , <i>bit_mask</i> , <i>v_adr</i> , <i>v_p1</i> , <i>v_p2</i> , <i>v_p3</i> ) <b>AC_TOGGLE_BIT</b> ( <i>ac_id</i> , <i>bit_mask</i> , <i>v_adr</i> , <i>v_p1</i> , <i>v_p2</i> , <i>v_p3</i> ) <u>Reaction element:</u> <b>RC_SET_BIT</b> ( <i>rc_id</i> , <i>ev_code</i> , <i>bit_mask</i> , <i>v_adr</i> , <i>v_p1</i> , <i>v_p2</i> , <i>v_p3</i> ) <b>RC_RESET_BIT</b> ( <i>rc_id</i> , <i>ev_code</i> , <i>bit_mask</i> , <i>v_adr</i> , <i>v_p1</i> , <i>v_p2</i> , <i>v_p3</i> ) <b>RC_TOGGLE_BIT</b> ( <i>rc_id</i> , <i>ev_code</i> , <i>bit_mask</i> , <i>v_adr</i> , <i>v_p1</i> , <i>v_p2</i> , <i>v_p3</i> )	

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>bit_mask</i>	Mask with bits to be set, reset or inverted. The mask is word-sized, i.e. up to 16 bits can be specified.
	<i>v_adr</i>	Data identifier for accessing a data in the NC/PLC or MMC (see Chapter 5).
	<i>v_p1 ... v_p3</i>	Additional parameters for data access (see Chapter 5).

## 4.2.2 SET\_BYTE, SET\_WORD, SET\_LONG, SET\_DOUBLE: Setting a value

**Description** These routines can be used to set constant values in format BYTE, UWORD, LONG or DOUBLE to a control system variable (NC/PLC or MMC) configured via parameters *v\_adr* and *v\_p1 ... v\_p3*.

**Syntax**

Action element:

**AC\_SET\_BYTE** (*ac\_id, wert\_w, v\_adr, v\_p1, v\_p2, v\_p3*)  
**AC\_SET\_WORD** (*ac\_id, wert\_w, v\_adr, v\_p1, v\_p2, v\_p3*)  
**AC\_SET\_LONG** (*ac\_id, wert\_l, v\_adr, v\_p1, v\_p2, v\_p3*)  
**AC\_SET\_DOUBLE** (*ac\_id, wert\_d, v\_adr, v\_p1, v\_p2, v\_p3*)

Reaction element:

**RC\_SET\_BYTE** (*rc\_id, ev\_code, wert\_w, v\_adr, v\_p1, v\_p2, v\_p3*)  
**RC\_SET\_WORD** (*rc\_id, ev\_code, wert\_w, v\_adr, v\_p1, v\_p2, v\_p3*)  
**RC\_SET\_LONG** (*rc\_id, ev\_code, wert\_l, v\_adr, v\_p1, v\_p2, v\_p3*)  
**RC\_SET\_DOUBLE** (*rc\_id, ev\_code, wert\_d, v\_adr, v\_p1, v\_p2, v\_p3*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>wert_w, wert_l, wert_d</i>	Constant value in the appropriate format (UWORD [e.g. 123], LONG [e.g. 33123L], DOUBLE [123.4567]), that is to be written into the specified data.
	<i>v_adr</i>	Data identifier for accessing a data in the NC/PLC or MMC (see Chapter 5).
	<i>v_p1 ... v_p3</i>	Additional parameters for data access (see Chapter 5).



### 4.2.3 CALC\_UWORD, CALC\_LONG, CALC\_DOUBLE: Calculating values

<b>Description</b>	These routines are used to perform arithmetic operations. In each operation, a constant value and a variable in format UWORD, LONG or DOUBLE as configured via parameters <i>v_adr</i> and <i>v_p1 ... v_p3</i> are computed in an arithmetic operation as defined in parameter <i>op</i> . The result of the computation is stored in the appropriate format in the configured variable in the control (NC/PLC or MMC).
<b>Syntax</b>	<p><u>Action element:</u>  <b>AC_CALC_UWORD</b> (<i>ac_id, wert_w, op, v_adr, v_p1, v_p2, v_p3</i>)  <b>AC_CALC_LONG</b> (<i>ac_id, wert_l, op, v_adr, v_p1, v_p2, v_p3</i>)  <b>AC_CALC_DOUBLE</b> (<i>ac_id, wert_d, op, v_adr, v_p1, v_p2, v_p3</i>)</p> <p><u>Reaction element:</u>  <b>RC_CALC_UWORD</b> (<i>rc_id, ev_code, wert_w, op, v_adr, v_p1, v_p2, v_p3</i>)  <b>RC_CALC_LONG</b> (<i>rc_id, ev_code, wert_l, op, v_adr, v_p1, v_p2, v_p3</i>)  <b>RC_CALC_DOUBLE</b> (<i>rc_id, ev_code, wert_d, op, v_adr, v_p1, v_p2, v_p3</i>)</p>
<b>Parameters</b>	<p><i>ac_id, rc_id</i> Unique identifier of the action or reaction element.</p> <p><i>ev_code</i> Code of event which must trigger processing of the reaction element.</p> <p><i>wert_w, wert_l, wert_d</i> Constant value in corresponding format (UWORD, LONG, DOUBLE e.g. 252, 111 000, 3567.345) to be computed with the specified data. If the calculated value must be displayed, the field must be converted in the same way as the calculation.</p> <p><i>op</i> Arithmetic operation which is applied to the constant value and the configured variable.</p> <p>AC_ADD Addition (variable = variable + value)</p> <p>AC_SUB Subtraction (variable = variable - value)</p> <p>AC_MUL Multiplication (variable = variable * value)</p> <p>AC_DIV Division (variable = variable / value)</p> <p>AC_MOD Modulo (variable = variable % value)</p> <p><i>v_adr</i> Data identifier for accessing a data in the NC/PLC or MMC (see Chapter 5).</p> <p><i>v_p1 ... v_p3</i> Additional parameters for data access (see Chapter 5).</p>

## 4.2.4 CALC\_DATA: Calculating with two variables

**Description** These routines are used to perform arithmetic operations on two configured variables. In this case, the configured variables are combined as specified in parameter *op*. The result of the calculation is stored in the desired format in the first of the configured NC/PLC/MMC variables.

**Syntax**

Action element:

**AC\_CALC\_DATA** (*ac\_id, v1\_adr, v1\_p1, v1\_p2, v1\_p3, op, v2\_adr, v2\_p1, v2\_p2, v2\_p3, d\_typ*)

Reaction element:

**RC\_CALC\_DATA** (*rc\_id, ev\_code, v1\_adr, v1\_p1, v1\_p2, v1\_p3, op, v2\_adr, v2\_p1, v2\_p2, v2\_p3, d\_typ*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>v1_adr</i>	Data identifier for accessing the first data in the NC/PLC/MMC (see Chapter 5). The result of the calculation is also stored in this data.
<i>v1_p1 ... v1_p3</i>	Additional parameters for accessing the first data (see Chapter 5).
<i>op</i>	Arithmetic operator representing operation to be performed on the two configured variables. AC_ADD Addition (variable1 = variable1 + variable2) AC_SUB Subtraction (variable1 = variable1 - variable2) AC_MUL Multiplication (variable1 = variable1 * variable2) AC_DIV Division (variable1 = variable1 / variable2) AC_MOD Modulo (variable1 = variable1 % variable2)
<i>v2_adr</i>	Data identifier for accessing the second data in the NC/PLC/MMC (see Chapter 5).
<i>v2_p1 ... v2_p3</i>	Additional parameters for accessing the second data (see Chapter 5).
<i>d_typ</i>	Data type; format in which the result of the arithmetic operation should be saved in the first configured variables. BTSS_CHAR       = char BTSS_UNSIGNED   = unsigned short int BTSS_WORD       = short int

BTSS\_LONG = long int  
BTSS\_DOUBLE = double (not with modulo, AC\_MOD)

If the calculated value must be displayed, the field must be converted in the same way as the calculation.

#### 4.2.5 COPY\_DATA: Copying a variable

<b>Description</b>	This routine copies the content of an NC/PLC/MMC variable to another variable.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_COPY_DATA</b> ( <i>ac_id, v1_adr, v1_p1, v1_p2, v1_p3, v2_adr, v2_p1, v2_p2, v2_p3</i> )  <u>Reaction element:</u> <b>RC_COPY_DATA</b> ( <i>rc_id, ev_code, v1_adr, v1_p1, v1_p2, v1_p3, v2_adr, v2_p1, v2_p2, v2_p3</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>v1_adr</i>	Data identifier for accessing the data source in the NC/PLC/MMC (see Chapter 5).
	<i>v1_p1 ... v1_p3</i>	Additional parameters for accessing the data source (see Chapter 5).
	<i>v2_adr</i>	Data identifier for accessing the data target in the NC/PLC/MMC (see Chapter 5).
	<i>v2_p1 ... v2_p3</i>	Additional parameters for accessing the data target (see Chapter 5).

#### 4.2.6 COPY\_DATA\_TO\_NB: Copying a variable into the notepad

<b>Description</b>	This routine copies the content of an NC/PLC/MMC variable to a notepad.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_COPY_DATA_TO_NB</b> ( <i>ac_id, v_adr, v_p1, v_p2, v_p3, nb_nr</i> )  <u>Reaction element:</u> <b>RC_COPY_DATA_TO_NB</b> ( <i>rc_id, ev_code, v_adr, v_p1, v_p2, v_p3, nb_nr</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.

<i>v_adr</i>	Data identifier for accessing the data source in the NC/PLC/MMC (see Chapter 5).
<i>v_p1 ... v_p3</i>	Additional parameters for accessing the data source (see Chapter 5).
<i>nb_nr</i>	Number of the notepad in which the value, read from the variables, should be written into.

### 4.2.7 COPY\_BLOCK\_NCK\_NB: Copying variables blockwise (OP 030)

**Description** This routine copies the content of NC/PLC/MMC variables in blocks to another area.

**Syntax**

Action element:  
**AC\_COPY\_BLOCK\_NCK\_NB** (*ac\_id, v1\_adr, v1\_p1, v1\_p2, v1\_p3, v2\_adr, v2\_p1, v2\_p2, v2\_p3, v3\_adr, v3\_p1, v3\_p2, v3\_p3*)

Reaction element:  
**RC\_COPY\_BLOCK\_NCK\_NB** (*ac\_id, ev-code, v1\_adr, v1\_p1, v1\_p2, v1\_p3, v2\_adr, v2\_p1, v2\_p2, v2\_p3, v3\_adr, v3\_p1, v3\_p2, v3\_p3*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>v1_adr</i>	Data identifier for accessing the data source in the NC/PLC/MMC (see Chapter 5).
<i>v1_p1 ... v1_p3</i>	Additional parameters for accessing the data source (see Chapter 5).
<i>v2_adr</i>	Data identifier for accessing the data target in the NC/PLC/MMC (see Chapter 5).
<i>v2_p1 ... v2_p3</i>	Additional parameters for accessing the data target (see Chapter 5).
<i>v3_adr</i>	Data identifier for accessing the variable in the NC/PLC/MMC in which the number of data to be copied is stored (see Chapter 5).
<i>v3_p1 ... v3_p3</i>	Additional parameters for accessing the data number (see Chapter 5).

## 4.2.8 CONVERT\_DATA\_FORMAT: Converting a data format

<b>Description</b>	This routine reads the content of an NC/PLC variable and stores the value in the desired data format in a notepad.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_CONVERT_DATA_FORMAT</b> ( <i>ac_id, v_adr, v_p1, v_p2, v_p3, qd_typ, nb_nr, zd_typ</i> )  <u>Reaction element:</u> <b>RC_CONVERT_DATA_FORMAT</b> ( <i>rc_id, ev_code, v_adr, v_p1, v_p2, v_p3, qd_typ, nb_nr, zd_typ</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>v_adr</i>	Data identifier for accessing the data source in the NC/PLC (see Chapter 5).
	<i>v_p1 ... v_p3</i>	Additional parameters for accessing the data source (see Chapter 5).
	<i>qd_typ</i>	Identifier for data type of data source; format in which value is read from the configured variable. Valid values are:  BTSS_CHAR               = char BTSS_UNSIGNED         = unsigned short int BTSS_LONG             = long int BTSS_DOUBLE          = double
	<i>nb_nr</i>	Number of notepad to which value that has been read and converted from the variable must be written.
	<i>zd_typ</i>	Identifier of target data type; format into which read value must be converted and saved in notepad <i>nb_nr</i> . The data identifiers here can be the same as for source data type (parameter <i>qd_typ</i> ).

## 4.3 General routines

### 4.3.1 CHANGE\_MODE: NC operating mode change) only OP 030)

**Description** This routine is used to change operating modes and designed specially for use on the SINUMERIK 840D/810D and FM-NC.

**Syntax**

Action element:  
**AC\_CHANGE\_MODE** (*ac\_id, modus*)

Reaction element:  
**RC\_CHANGE\_MODE** (*rc\_id, ev\_code, modus*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>modus</i>	Mode BA_JOG JOG mode BA_MDA MDA mode BA_AUTOMATIC AUTOMATIC mode BA_JOG_REFPOINT REFPOINT mode BA_JOG_REPOS REPOS mode BA_JOG_PRESET PRESET mode

### 4.3.2 CHANNEL\_SWITCH: Changing the NC channel

**Description** This routine switches the screen displays to the next possible NC channel. All active windows are closed and reopened again for both the global and the local menu.

**OP 030 only:**  
The number of the active channel is contained in notepad entry NB\_ACT\_CHANNEL in format F\_UWORD. It is refreshed by this routine.

<b>Syntax</b>	<u>Action element:</u>	
	<b>AC_CHANNEL_SWITCH</b> ( <i>ac_id</i> )	
	<u>Reaction element:</u>	
	<b>RC_CHANNEL_SWITCH</b> ( <i>rc_id, ev_code</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.

### 4.3.3 **BV\_LANGUAGE\_CHANGE: Toggling between languages (OP 030)**

<b>Description</b>	This routine switches the operator interface between the background and foreground languages, refreshing the current screen contents at the same time. The number of the newly selected language is returned to the specified notepad.	
<b>Syntax</b>	<u>Action element:</u>	
	<b>AC_BV_LANGUAGE_CHANGE</b> ( <i>ac_id, nb_nr</i> )	
	<u>Reaction element:</u>	
	<b>RC_BV_LANGUAGE_CHANGE</b> ( <i>rc_id, ev_code, nb_nr</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>nb_nr</i>	Number of notepad to which number of selected language must be written.

### 4.3.4 **SET\_MESSAGE, RESET\_MESSAGE: Setting, resetting a message**

<b>Description</b>	These routines are used to set or reset MMC messages that are displayed in the dialog line and suitable for use as operator guides. The texts displayed with SET_MESSAGE in the dialog line must be stored in text files <b>ALA.TXT</b> (OP 030) and <b>ALM.TXT</b> (MMC 100/UOP).	
<b>Syntax</b>	<u>Action element:</u>	
	OP 030: <b>AC_SET_MESSAGE</b> ( <i>ac_id, msg_nr</i> )	
	MMC 100/UOP: <b>AC_SET_MESSAGE</b> ( <i>ac_id, msg_nr, 0, 0</i> ) <b>AC_RESET_MESSAGE</b> ( <i>ac_id</i> )	

Reaction element:

OP 030:

**RC\_SET\_MESSAGE** (*rc\_id, ev\_code, msg\_nr*)

MMC 100/UOP:

**RC\_SET\_MESSAGE** (*rc\_id, ev\_code, msg\_nr, 0, 0*)

**RC\_RESET\_MESSAGE** (*rc\_id, ev\_code*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code e</i>	Code of event which must trigger processing of the reaction element.
	<i>msg_nr</i>	Number of message to be set or reset. This must be within the numerical range  (120000...129999), file ALA.TXT for the OP 030 and (119000...119999), file ALM.TXT for the MMC 100/UOP

### 4.3.5 SET\_MSG\_POS: Setting the position of the message line (MMC100/EBF)

**Description** This routine is used to set the position and character length of the message line. The default position is above the soft key line, stretching across the entire width of the screen.

**Syntax**

Action element:

**AC\_SET\_MSG\_POS** (*ac\_id, x, y, len*)

Reaction element:

**RC\_SET\_MSG\_POS** (*rc\_id, ev\_code, x, y, len*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>x, y</i>	Absolute position of top, left-hand corner of message line in pixels as measured from the physical screen zero point
	<i>len</i>	Number of characters in message line (one character is 8 pixels wide, the entire screen 640 pixels)



### 4.3.6 SET\_CUR\_TXT\_POS: Setting the position of the cursor text

<b>Description</b>	This routine is used to set the position, character length and foreground and background colors of the cursor text line. (Cursor appears when the cursor is moved to a dialog field for which a cursor text is configured). The default position and the standard length are the values set with SET_MSG_POS. The default foreground color is yellow, the default background color black.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_SET_MSG_POS</b> ( <i>ac_id, x, y, len, fc, bc</i> )  <u>Reaction element:</u> <b>RC_SET_MSG_POS</b> ( <i>rc_id, ev_code, x, y, len, fc, bc</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>x, y</i>	Absolute position of top, left-hand corner of message line in pixels as measured from the physical screen zero point
	<i>len</i>	Number of characters in message line (one character is 8 pixels wide, the entire screen 640 pixels) Parameter <i>len</i> can be OR'd bit-serially with attribute <b>CT_CENTRE_ADJUST</b> or <b>CT_RIGHT_ADJUST</b> for centered or right-justified cursor text output.
	<i>fc</i>	Foreground color (refer to Section 3.3).
	<i>bc</i>	Background color (refer to Section 3.3).

### 4.3.7 SET\_INFO, RESET\_INFO: Output, delete help symbol

<b>Description</b>	These routines show or delete the Help symbol. The display positions are predefined.  The Info symbol (i) indicates that you can call additional information about the current dialog.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_SET_INFO</b> ( <i>ac_id</i> ) <b>AC_RESET_INFO</b> ( <i>ac_id</i> )  <u>Reaction element:</u> <b>RC_SET_INFO</b> ( <i>rc_id, ev_code</i> ) <b>RC_RESET_INFO</b> ( <i>rc_id, ev_code</i> )	

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.

**Note** The SET\_INFO and RESET\_INFO routines are not available for OP 030!

### 4.3.8 TOOL\_SEARCH: Search for tool in the actual TO area (OP 030)

**Description** This routines searches for the parameterized tool in the current TO area and writes the search result to the specified notepads.

**Syntax**

Action element:  
**AC\_TOOL\_SEARCH** (*ac\_id, nb\_tool, nb\_test, nb\_index*)

Reaction element:  
**RC\_TOOL\_SEARCH** (*rc\_id, event, nb\_tool, nb\_test, nb\_index*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>nb_tool</i>	Notepad number from which the routine reads the number of the tool to be searched for.
	<i>nb_test</i>	Notepad number in which the routine returns the search result. 0 = tool exists, 1 = tool does not exist.
	<i>nb_index</i>	Notepad number to which routine returns the column index of the search tool if it exists. The returned value is 0 if the tool cannot be found.

### 4.3.9 TOOL\_CREATE: Create new tool (OP 030)

**Description** This routine creates a new tool with the first cutting edge, or sets up a new edge.

**Syntax**

Action element:  
**AC\_TOOL\_CREATE** (*ac\_id, nb\_tool, nb\_to, mode*)

Reaction element:  
**RC\_TOOL\_SEARCH** (*rc\_id, event, nb\_tool, nb\_to, mode*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>nb_tool</i>	Notepad number from which routine reads the number of tool to be created.
	<i>nb_to</i>	Notepad number from which the routine reads the TO area.
	<i>mode</i>	Mode WZ_TOOL_CREATE Create tool and 1st cutting edge. (Tool must not exist already. (TOOL_SEARCH)) WZ_EDGE_CREATE Another cutting edge of the tool is set up. (Tool must already exist. (TOOL_SEARCH))

#### 4.3.10 TOOL\_DELETE: Deleting a tool (OP 030)

**Description** This routine deletes an existing tool.

**Syntax**

Action element:  
**AC\_TOOL\_DELETE**(*ac\_id, nb\_tool, nb\_to*)

Reaction element:  
**RC\_TOOL\_SEARCH** (*rc\_id, event, nb\_tool, nb\_to*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>nb_tool</i>	Notepad number from which routine reads the number of tool to be deleted.
	<i>nb_to</i>	Notepad number from which the routine reads the TO area.

## 4.4 Routines to handle part programs

### 4.4.1 PP\_EDIT\_OPEN: Opening an edit field for part programs

**Description** This routine opens an edit field (editor) in a window. A corresponding EDIT\_FIELD must be configured in the object list for the window. You can only open one edit field in each window.

**Syntax**

Action element:  
**AC\_PP\_EDIT\_OPEN** (*ac\_id, menu\_typ, win\_id, ef\_id*)

Reaction element:  
**RC\_PP\_EDIT\_OPEN** (*rc\_id, ev\_code, menu\_typ, win\_id, ef\_id*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>menu_typ</i>	Menu type GLOBAL The edit field must be opened in a window of the global menu. LOCAL The edit field must be opened in a window of the local menu.
<i>win_id</i>	Identifier of the window in which the edit field must be opened.
<i>ef_id</i>	Identifier of the edit field to be opened.

**Note** PP\_EDIT\_OPEN must be called in the open list (OPEN\_LIST) of a menu after the window to which the edit field belongs has been opened. You can do this by parameterizing the window of the edit field with attribute W\_OPEN\_AFTER\_OBJ.

#### 4.4.2 PP\_EDIT\_CLOSE: Closing an edit field for part programs

<b>Description</b>	This routine closes an edit field (editor) in a window.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_PP_EDIT_CLOSE</b> ( <i>ac_id, menu_typ, win_id</i> )  <u>Reaction element:</u> <b>RC_PP_EDIT_CLOSE</b> ( <i>rc_id, ev_code, menu_typ, win_id</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>menu_typ</i>	Menu type GLOBAL The edit field must be closed in a window of the global menu. LOCAL The edit field must be closed in a window of the local menu.
	<i>win_id</i>	Identifier of window in which edit field must be closed.
<b>Note</b>	PP_EDIT_CLOSE must be called in the close list (CLOSE_LIST) of the window to which the edit field belongs.	

#### 4.4.3 PP\_REFRESH: Refreshing an edit field for part programs

<b>Description</b>	This function refreshes the content of an edit field for part programs or other assigned data.  The content of the data to be displayed is read again.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_PP_REFRESH</b> ( <i>ac_id, win_id</i> )  <u>Reaction element:</u> <b>RC_PP_REFRESH</b> ( <i>rc_id, ev_code, win_id</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>win_id</i>	Identifier of window in which edit field must be refreshed.

## 4.5 Routines for diagnostics functionality

### 4.5.1 DG\_INIT\_ALARM: Initializing for the alarm overview (OP 030)

**Description** This routine initializes the display of alarms (NCK alarms, MMC alarms and PLC alarms). The alarm display is structured in an internal list in such a way that each alarm (250 characters maximum) is output on 7 lines. In this case, the first two lines are each 26 characters in length (to leave room to display the cancel criterion); the following 4 lines are each 29 characters in length and the rest of the text (250 characters in total) is stored on the last line.

**Syntax**

Action element:  
**AC\_DG\_INIT\_ALARM** (*ac\_id, fst\_visu\_nb, page\_nb, cnt\_nb, info\_nb*)

Reaction element:  
**RC\_DG\_INIT\_ALARM** (*rc\_id, ev\_code, fst\_visu\_nb, page\_nb, cnt\_nb, info\_nb*)

<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>fst_visu_nb</i>	First of seven sequential notepads selected to visualize the seven lines (one notepad is needed to display each line in the alarm overview). These are allocated to the seven lines of the internal alarm text list by the initialization routine, and thus each contain one line of the alarm text in format CON_STRING.
	<i>page_nb</i>	Notepad needed to page within the alarm overview. This specifies in each case the alarm to be displayed from the list. The index to the alarm list is specified in data format UWORD.
	<i>cnt_nb</i>	Notepad in which the current number of alarms (in format UWORD) is stored. The number can be viewed in an O_FIELD.
	<i>info_nb</i>	Notepad in which the cancel criterion information relating to the alarm being viewed is stored. The value represents the number of an object list containing the cancel criterion as a symbol. This data is stored in format UWORD and can be used for a PICT_FIELD.

The notepad can contain the values listed below:

OB\_POLY\_POWER\_ON (for Power-on alarm),  
 OB\_POLY\_RESET (for Reset alarms),  
 OB\_POLY\_CANCEL (for Cancel alarms),  
 OB\_POLY\_NC\_START (for NC Start alarm),  
 OB\_POLY\_KEY (for Big-Mac alarm key or soft key),  
 OB\_POLY\_EMPTY (no cancel criterion).

## 4.5.2 DG\_INIT\_MSG: Initializing for the message overview

<b>Description</b>	This routine initializes the display of PLC messages. The message display is structured in an internal list in such a way that each message (250 characters maximum) is output on 7 lines. In this case, the first two lines are each 26 characters in length (to leave room to display the cancel criterion); the following 4 lines are each 29 characters in length and the rest of the text (250 characters in total) is stored on the last line.	
<b>Syntax</b>	<u>Action element:</u> <b>AC_DG_INIT_MSG</b> ( <i>ac_id, fst_visu_nb, page_nb, cnt_nb, info_nb</i> )  <u>Reaction element:</u> <b>RC_DG_INIT_MSG</b> ( <i>rc_id, ev_code, fst_visu_nb, page_nb, cnt_nb, info_nb</i> )	
<b>Parameters</b>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
	<i>fst_visu_nb</i>	First of seven sequential notepads selected to visualize the seven lines (one notepad is needed to display each line in the message overview). These are allocated to the seven lines of the internal message text list by the initialization routine, and thus each contain one line of the message text in format CON_STRING.
	<i>page_nb</i>	Notepad needed to page within the message overview. This specifies in each case the message to be displayed from the message list. The index to the message list is specified in data format UWORD.
	<i>cnt_nb</i>	Notepad in which the current number of messages (in format UWORD) is stored. The number can be viewed in an O_FIELD.
	<i>info_nb</i>	Notepad in which the cancel criterion information relating to the message being viewed is stored. The value represents the number of an object list containing the cancel criterion as a symbol. This data is stored in format UWORD and can be used for a PICT_FIELD.

The notepad can contain the values listed below:

OB\_POLY\_POWER\_ON (for Power-On alarm),  
OB\_POLY\_RESET (for Reset alarms),  
OB\_POLY\_CANCEL (for Cancel alarms),  
OB\_POLY\_NC\_START (for NC Start alarm),  
OB\_POLY\_KEY (for Big-Mac alarm key or soft key),  
OB\_POLY\_EMPTY (no cancel criterion; normal for messages).

### 4.5.3 OPEN\_VERSION: Initializing for the NCK version display

**Description** This routine initializes the NCK version display. It must be called within the OPEN\_LIST of the window in which the NCK version file must be viewed.



---

**Important**

Since this routine uses up dynamic memory, the CLOSE\_VERSION routine must be called in the CLOSE\_LIST of the appropriate window to free up this dynamic memory again.

---

**Syntax** Action element:  
**AC\_OPEN\_VERSION** (*ac\_id, fname\_nb*)  
Reaction element:  
**RC\_OPEN\_VERSION** (*rc\_id, ev\_code, fname\_nb*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>fname_nb</i>	First of two sequential notepads to which the routine writes the file name of the NCK version file for an EDIT_FIELD.



#### 4.5.4 CLOSE\_VERSION: Closing the NCK version display

**Description** This routine ends the display of the NCK version file and releases the dynamic memory which has been requested by the OPEN\_VERSION routine. It must be called in the CLOSE\_LIST of the window in which the NCK version is displayed.

**Syntax**

Action element:  
**AC\_CLOSE\_VERSION** (*ac\_id, fname\_nb*)

Reaction element:  
**RC\_CLOSE\_VERSION** (*rc\_id, ev\_code, fname\_nb*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>fname_nb</i>	First of two sequential notepads. This number must be the same as the number passed to the OPEN_VERSION routine.

#### 4.5.5 DG\_INIT\_PASSW: Initializing the password dialog

**Description** This routine initializes the dialog for processing the password (set password, change password, deactivate a set password). It also requests dynamic memory.

**Syntax**

Action element:  
**AC\_INIT\_PASSW** (*ac\_id, passw\_nb*)

Reaction element:  
**RC\_INIT\_PASSW** (*rc\_id, ev\_code, passw\_nb*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>passw_nb</i>	Number of notepad to be used to enter the password via an input/output field.

### 4.5.6 DG\_CLOSE\_PASSW: Closing the password dialog

**Description** This routine ends the password dialog in a defined manner, also releasing the dynamic memory again that was requested by the 'dg\_init\_passw' routine. It must be called within the CLOSE\_LIST for the password dialog.

**Syntax**

Action element:  
**AC\_CLOSE\_PASSW** (*ac\_id*)

Reaction element:  
**RC\_CLOSE\_PASSW** (*rc\_id, ev\_code*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.

### 4.5.7 DG\_SET\_PASSW: Setting the password (OP 030)

**Description** This routine uses a PI service to send the password, stored in notepad 'passw\_nb' to the NC kernel. It may only be called if the 'DG\_INIT\_PASSW' routine, which specifies the number of the notepad for password entry ('passw\_nb'), has already been called. The result of the routine is stored in notepad 'ret\_nb' and can be subsequently used again in the action or reaction list.

**Syntax**

Action element:  
**AC\_SET\_PASSW** (*ac\_id, ret\_nb*)

Reaction element:  
**RC\_SET\_PASSW** (*rc\_id, ev\_code, ret\_nb*)

**Parameters**

<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.
<i>ret_nb</i>	Number of notepad in which the return value of the routine must be stored. The content of this notepad has the following meaning after the routine has been called: 0 ---> Set password 1 ---> Error; password not set

### 4.5.8 DG\_CHG\_PASSW: Changing the password for the actual access stage

<b>Description</b>	<p>This routine changes the password for the current access level, i.e. the password for the desired access level must already be set (using DG_SET_PASSW) before the routine is called. It may be called only after the password dialog has been initialized by DG_INIT_PASSW. When the routine is first called, it notes the password stored in notepad '<i>passw_nb</i>' (specified by the DG_INIT_PASSW routine). The return value of the routine (notepad '<i>ret_nb</i>') is not relevant in this case. When the routine is called a second time (CONFIRM PASSWORD function), it checks whether the password entered beforehand is the same as the password which has been re-entered in '<i>passw_nb</i>' and, if it is, changes the password to the NC kernel using a PI service.</p>						
<b>Syntax</b>	<p><u>Action element:</u>  <b>AC_CHG_PASSW</b> (<i>ac_id, ret_nb</i>)</p> <p><u>Reaction element:</u>  <b>RC_CHG_PASSW</b> (<i>rc_id, ev_code, ret_nb</i>)</p>						
<b>Parameters</b>	<table> <tr> <td style="padding-right: 20px;"><i>ac_id, rc_id</i></td> <td>Unique identifier of the action or reaction element.</td> </tr> <tr> <td><i>ev_code</i></td> <td>Code of event which must trigger processing of the reaction element.</td> </tr> <tr> <td><i>ret_nb</i></td> <td>Number of notepad in which the return value of the routine must be stored. The content of this notepad has the following meaning after every second routine call (CONFIRM PASSWORD function):            0 ---&gt; Password for current access level changed            1 ---&gt; Passwords do not match.</td> </tr> </table>	<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.	<i>ev_code</i>	Code of event which must trigger processing of the reaction element.	<i>ret_nb</i>	Number of notepad in which the return value of the routine must be stored. The content of this notepad has the following meaning after every second routine call (CONFIRM PASSWORD function): 0 ---> Password for current access level changed 1 ---> Passwords do not match.
<i>ac_id, rc_id</i>	Unique identifier of the action or reaction element.						
<i>ev_code</i>	Code of event which must trigger processing of the reaction element.						
<i>ret_nb</i>	Number of notepad in which the return value of the routine must be stored. The content of this notepad has the following meaning after every second routine call (CONFIRM PASSWORD function): 0 ---> Password for current access level changed 1 ---> Passwords do not match.						



FiIDocStart

## Notes

# 5

## 5 Accessing Data on NCK / PLC / MMC

5.1 NCK variables – address structure .....	5-198
5.2 NCK variable - configuring .....	5-201
5.3 PLC variable – address structure .....	5-204
5.4 PLC variable - configuring .....	5-206
5.5 MMC-variable (notepad entry – address structure .....	5-209
5.6 MMC variable (notepad entry) - configuring .....	5-210
5.7 Access to variables - example .....	5-210

The three variable types listed can be used with all dynamic display elements and computation or copy routines.  
 The display elements represent the variables on the operator interface.

## 5.1 NCK variables – address structure

Variables are organized in the NCK according to the following basic principle:

- The NCK has several areas.
- Each area has several blocks.
- Each block has several variables.
- Variables are arranged two-dimensionally in a block  
 -> Columns and rows.
- Each variable in the block can be addressed uniquely on the basis of column and row.
- Each column has its own data type. This means:  
 Variables, that are located in one column (the same column index, but different row index), always have the same data type.

Unique addressing of variables in the 840D and FM-NC is based on 8 bytes, which clearly define the variable in the NCK.

<b>Address structure for NCK variables</b>	15	7	0
	Syntax ID =		Area =                      Unit =
	Column index =		
	Row index =		
	Block type =		No. of rows =

**Syntax ID**                      **NUMERIK8** identifier for NCK variables

**Area**                              The area specifies the location of the variable:

- N**            Global NCK area
- B**            Mode group area
- C**            Channel area
- A**            Axis area
- T**            Tool data area
- V**            Feed drive area
- H**            Main spindle drive area

The identifiers listed above are symbolic defines only and are replaced by values in the real variable address.

**Unit** The unit specifically defines the area. The variable address includes 5 bits for the unit, allowing units 1-31 to be addressed (value 0 is reserved).

e.g.: area C, unit 1 => first channel  
area A, unit 3 => third axis

**Block type**

Sub-area within an area.

<b>FU</b>	User frame
<b>FA</b>	Actual frame
<b>S</b>	Status data
<b>SINF</b>	Status data, channel influences
<b>SPARP</b>	Status data, part program information
<b>SNCF</b>	Status data, NC functions
<b>SPARPP</b>	Status data "program pointer"
<b>SPARPF</b>	Status data "search pointer"
<b>SSYNAC</b>	Status data "synchronized actions"
<b>SALA</b>	Status data, alarms
<b>SALAP</b>	Status data, oldest alarm
<b>SALAL</b>	Status data, most recent alarm
<b>SMA</b>	Status data, machine axes
<b>SGA</b>	Status data, geometrical axes
<b>SSP</b>	Status data, spindles
<b>SEMA</b>	Extended status data, machine axes
<b>SEGA</b>	Extended status data, geometrical axes
<b>SPARPI</b>	Program pointer for the "program interrupted" status
<b>Y</b>	System data
<b>YNCFL</b>	NC instruction group list
<b>TO</b>	Tool offsets
<b>TV</b>	Tool directory
<b>TD</b>	General tool data
<b>TS</b>	Tool monitoring data
<b>TU</b>	User-specific tool data
<b>TUE</b>	User-specific tool cutting data
<b>TG</b>	Grinding-specific tool data
<b>TMV</b>	Magazine directory
<b>TMC</b>	Magazine configuration data
<b>TM</b>	Magazine data
<b>TT</b>	Magazine location types
<b>TPM</b>	Multiple location assignments
<b>TP</b>	Magazine location data

<b>PA</b>	Protection areas
<b>RP</b>	Arithmetic parameters
<b>SE</b>	Setting data
<b>GUD</b>	Global user data
<b>LUD</b>	Local part program data
<b>M</b>	Machine Data

The identifiers listed above are symbolic defines only and are replaced by values in the real variable address.

**Column index**  
**Line index**

Unique addressing of variable within a block.

The row index is used in many block types to provide more accurate definition through additional indexing:

**References:** /LIS/, Lists

**Number of rows**

Access via a variable address to several values in a column.

For individual descriptions of blocks and variables, please see

**References:** /LIS/, Lists



## 5.2 NCK variable - configuring

**NC variable definition** Find the variable that you wish to use in document  
**OPI - btss\_var.h**

**References:** /LIS/, Lists

**Include-File** *btss\_var.h*.

This list specifies the variable name, the block type and the area for the variable.

To facilitate access to the variables, the include file *btss\_var.h* mentioned above contains predefined symbolic NC variable addresses.

These predefined symbolic NC variable addresses are structured as follows:

**P\_<Area>\_<Block type>\_<Variable name>**

### Example

Configure the variable "**Channel status**"

Area **C** (channel)

Block type **S** (status variable)

Variable name **chanStatus**

=> symbolic NC variable address **P\_C\_S\_chanStatus**

A symbolic NC variable address contains the following information about a variable:

- **Syntax ID**
- **Area**
- **Block type**
- **Column index**
- **Data type (variable format)**

---

#### Note

The number of data to be read (number of rows) is always preset to "1" when the configuration is used.

---

### Data types of NC variables

Data type	Bits	Value range
BTSS_CHAR	8	0..255
BTSS_UNSIGNED	16	0..65535
BTSS_WORD	16	-32767..32767
BTSS_LONG	32	-2.147.483.648 .. +2.147.483.648

BTSS_FLOAT	32	+/- 10 <sup>-037</sup> .. +/- 10 <sup>+038</sup>
BTSS_DOUBLE	64	+/- 10 <sup>-307</sup> .. +/- 10 <sup>+308</sup>
BTSS_STRING_1 .. 100		
BTSS_STRING_<string length>	8* String length	ASCII string of length "String length" (1-100) incl. "\0"
BTSS_STRING_128	8*128	ASCII string with 128 characters incl. "\0" (for NC messages)
BTSS_STRING_198	8*198	ASCII string with 198 characters incl. "\0" (for NC single blocks)

**Additional parameters  
for NC variables**

Apart from the information already included in symbolic variable addresses, you can also change the following data when you configure them:

- **Row index,**
- **Unit**
- **Column index**

This information is manipulated by three additional variable-accessing parameters.

- These are designated below as **Par1**, **Par2** and **Par3**. Please note when configuring addresses that the parameters must be specified in the order Par1, Par2, Par3.
- Each addition parameter is 2 bytes in length.
- Parameter **Par2** determines the meaning and what influence the values from Par1 and Par3 have on the variable parts (row index, column index and unit) of the NC variable address.

**Meaning of additional parameters for NC variables:**

Par1	Par2	Par3	Meaning
<Row index>		<Unit>	<p><b>Row index</b> = &lt;Row index&gt;</p> <p><b>Unit</b> = &lt;Unit&gt;</p> <p>If the unit = 0 and the area = C, then <b>Unit</b> is set to the current channel; when area = B, it is set to the current mode group.</p>

<Notepad number>	<b>NB_ROW</b>	<Offset>	<b>Row index</b> = (contents from <Notepad number>) + <Offset>  If the unit = 0 and the area = C, then <b>Unit</b> is set to the current channel; when area = B, it is set to the current mode group.
<Notepad number>	<b>NB_UNIT</b>	<Offset>	<b>Row index</b> = 1  <b>Unit</b> = (contents from <Notepad number>) + <Offset>
<Notepad number1>	<b>NB_ROW_UNIT</b>	<Notepad number2>	<b>Row index</b> = contents from <Notepad number1>  <b>Unit</b> = contents from <Notepad number2>
1st byte: <Notepad number1> (max. 255)  2nd byte: <Notepad number2> (max. 255)	<b>NB_COLUMN_ROW</b>	<Offset>	<b>Column index</b> = contents from <Notepad number1>  <b>Row index</b> = contents from <Notepad number2> + <Offset>  If the unit = 0 and the area = C, then <b>Unit</b> is set to the current channel; when area = B, it is set to the current mode group.  <b>Example:</b> P_T_TO_cuttEdgeParam, (UWORD)(NB_PNo_row<<8 NB_TNo_column), NB_COLUMN_ROW, Offset_row

Par1	Par2	Par3	Meaning
<Row index>	<b>COLUMN_OFFSET</b>	<Offset>	<b>Row index</b> = <Row index>  <b>Column index</b> = column index (from variable definition) + <Offset>  If the unit = 0 and the area = C, then <b>Unit</b> is set to the current channel; when area = B, it is set to the current mode group.
1st byte: <Notepad number1> (max. 255)  2nd byte: <Notepad number2> (max. 255)	<b>NB_ROW_UNIT_OFFSET_ROW</b>	<Offset>	<b>Row index</b> = (content from <Notepad number1>)+ <Offset>  <b>Unit</b> = (contents from <Notepad number2>)  <b>Example:</b> P_T_TV_toolNo, NB_T_unit<<8 NB_T_row, NB_ROW_UNIT_OFFSET_ROW, Offset_row

## 5.3 PLC variable – address structure

**Address structure for PLC variables** The variable address for the PLC comprises 10 bytes, which uniquely define the variable within the PLC.

15	7
Syntax ID =	Type =
Number =	
Sub-area =	
Area =	Offset =

**Syntax ID** **PLC10** Identifier for PLC variable

Type	Data type	Bits	Value range
	<b>BTSS_S7_BOOL</b>	8	0 or 1 in LSB
	<b>BTSS_CHAR</b>	8	0 ... 255
	<b>BTSS_UNSIGNED</b>	16	0 ... 65535
	<b>BTSS_S7_16BIT</b>	16	Bit string
	<b>BTSS_S7_32BIT</b>	32	Bit string
	<b>BTSS_LONG</b>	32	- 2 147 483 647 ... + 2 147 483 647
	<b>BTSS_S7_REAL</b>	64 (on OP 030) 32 (on PLC) converted during reading and writing	
	<b>BTSS_S7_COUNTER</b>	16	0.. 65535 or -32767 .. + 32767
	<b>BTSS_S7_TIMER</b>	16	0.. 65535 or -32767 .. + 32767

**Number** Number of variables that are addressed via this address.

**Area** The area specifies the location of the variable:

- PLC\_DB** Data block
- PLC\_DS** Data record
- PLC\_E** Input
- PLC\_A** Output
- PLC\_M** Flag
- PLC\_S7\_COUNTER** Counter
- PLC\_S7\_TIMER** Timer

The identifiers listed above are symbolic defines only and are replaced by values in the real variable address.

<b>Sub-area</b>	DB number	for area PLC_DB
	Block number	for area PLC_DS
<b>Offset</b>	Offset in bits	for area PLC_DB, PLC_E, PLC_A, PLC_M,
	No. of function element	for area PLC_S7_COUNTER, PLC_S7_TIMER
	Logical module address	for area PLC_DS

## 5.4 PLC variable - configuring

### Standard PLC variable definitions for SINUMERIK 840D/FM-NC - plc\_var.h

Include file "plc\_var.h" contains predefined symbolic PLC variable addresses for the standard PLC variables of a SINUMERIK 840D/FM-NC.

Variables for the following main groups can be accessed:

#### NC-specific signals:

- Handwheel selection

#### Mode-group-specific signals:

- Operating mode selection
- Machine function selection

#### Channel-specific signals:

- Program control
- Increment selection for geometry axes

#### Machine-axis-specific signals:

- Increment selection for machine axes

**References:** /FB/, A2, "Various Interface Signals"

### Definition of new symbolic PLC variable addresses

Further symbolic PLC variable addresses must be defined in the following form as a preprocessor macro for the specific application in each case:

```
#define <P_PLC_variable>          PLC10, <Type>, <sub-area>, \  
                                  (UWORD)(< Area> << 8 | \  
                                  <Offset bit23-bit16>), <Offset bit15-bit0>
```

"\" is the identifier for the preprocessor macro continuation in the next line

### Example

Symbolic PLC variable address for the "NCK battery alarm" and "NCK air temperature alarm" signals:

```
#define P_PLC_N_S_batt_alarm      PLC10, BTSS_S7_BOOL, 10,\  
                                  (UWORD)(PLC_DB << 8),109*8+7  
  
#define P_PLC_N_S_luft_alarm     PLC10, BTSS_S7_BOOL, 10,\  
                                  (UWORD)(PLC_DB << 8),109*8+6
```

Other defines can be used for the direct values:

```
#define DB_AREA_NC 10  
#define NC_DBX_ALARM 109*8
```

```
#define P_PLC_N_S_batt_alarm
                PLC10, BTSS_S7_BOOL, \DB_AREA_NC, \
                (UWORD)(PLC_DB << 8), NC_DB_ALARM+7

#define P_PLC_N_S_luft_alarm
                PLC10, BTSS_S7_BOOL, \DB_AREA_NC, \
                (UWORD)(PLC_DB << 8), NC_DBX_ALARM+6
```

For more information about preprocessor macros, see C syntax help in programmers environment:

The symbolic PLC variable address thus contains the following information about a PLC variable:

- **Syntax ID,**
- **Type,**
- **Sub-area,**
- **Area and**
- **Offset**

---

#### Note

The number is always preset to " 1" when used from the configuring level.

---

#### Additional parameters for PLC variable access

The following information can also be changed at the configuring stage:

- **Sub-area** (absolute and additive)
- **Offset** (absolute and additive)

This information is manipulated by three additional parameters for accessing PLC variables.

- These are designated below as **Par1**, **Par2** and **Par3**. Please note when configuring addresses that the parameters must be specified in the order Par1, Par2, Par3.
- Each addition parameter is 2 bytes in length.
- Parameter **Par2** determines the meaning and what influence the values from Par1 and Par3 have on the variable parts (sub-area and offset) of the PLC variable address.

#### Meaning of additional parameters for PLC variables:

- The sub-area can be changed via Par1 and the offset via Par3.
- It is possible to combine both by combining the appropriate defines in Par2 (e.g. Par2: PLC\_TB|PLC\_OF\_NB).
- Symbolic PLC variable addresses can be used subsequently in exactly the same way as NCK variable addresses. Explicit reference is made to any differences between the two address types.

**Modification of sub-area**

Par1	Par2	Meaning
0	0	No modification of <b>sub-area</b>
<Sub-area>	PLC_SA	Set <b>sub-area</b> with <Sub-area>
<Notepad number>	PLC_SA_NB	Set <b>sub-area</b> with content from <notepad number>
<Sub-area offset>	PLC_SA_AD	Add <sub-area offset> to the <b>sub-area</b>
<Notepad number>	PLC_SA_NB_AD	Add the contents from <Notepad number> to the <b>sub-area</b>
<Channel group>	PLC_SA_CH_GR	Add the actual channel number of the <Channel group> to the sub-area
0	PLC_SA_ACT_CHANN	Add the actual channel number of the <Channel group> to the sub-area

**Modification of offset**

Par2	Par3	
0	0	No modification of <b>offset</b>
PLC_OF	<Offset>	Set <b>offset</b> with <Offset> (max. 65535 - 2 bytes)
PLC_OF_NB	<Notepad number>	Set <b>offset</b> with content from <Notepad number> (3 bytes)
PLC_OF_AD	<DeltaOffset>	Add <DeltaOffset> to the <b>Offset</b> (max. 65535 – 2byte)
PLC_OF_NB_AD	<Notepad number>	Add the contents from <Notepad number> to the <b>Offset</b> (3 bytes)



## 5.5 MMC-variable (notepad entry – address structure)

### Notepad entries

MMC variables that can be used internally in the OP 030 and MMC 100/UOP are called "notepad" entries.

A notepad is a memory area of 8 bytes, which is identified by a unique number. One data type per notepad is also managed by the configuring functions. This is defined when variables are saved to the notepad. The data type cannot be addressed explicitly, but must be used correctly by linkage functions.

Notepad entries are internal variables of 8 bytes in size.

Using copying and arithmetic functions, it is possible to assign to these variables a

- Bit
- Byte
- Word (two bytes)
- Long (four bytes)
- Double (eight bytes)
- Pointer (four bytes), e.g. pointer to strings or texts

The notepad entries are organized into different areas:

#### OP 030:

- Local notepad entries
- Global notepad entries for OP 030 system
- Global notepad entries for standard configuration
- Global notepad entries for application configuration

#### MMC 100/UOP:

- Local notepad entries
- Global notepad entries for one operating area application

Local notepad entries should be used only temporarily within one list.

Global notepad entries can be used in all lists.

The areas are subdivided according to the notepad entry number:

0	..	99	local, i.e. as soon as the application is exited, data is rejected.
200	..	400	global, i.e. data are kept – also after changing-over the operator area

### Notepad entries

Preprocessor definitions can be assigned for notepad numbers.

The application area for notepad entries is defined in

**References:** /FBO/, EU, Development Kit  
/PJE/, HMI Embedded Configuring Package

## 5.6 MMC variable (notepad entry) - configuring

**Notepad entries** Notepad entries are identified by **P\_NB** in the configuration.

**Additional parameters for notepads** The content of the notepad entry can be manipulated further via **additional parameters for notepads**.

- These are designated below as **Par1**, **Par2** and **Par3**.
- Each addition parameter is 2 bytes in length.
- **Par2 determines the meaning and what influence the values from Par1 and Par3 have on the notepad entry.**

**Meaning of additional parameters for a notepad entry:**

Par1	Par2	Par3	Meaning
<NB_nr>	0	<Offset>	<b>Value</b> = (content from <i>Notepad</i> [NB_nr]) + <Offset> (the offset format is UWORD)
<NB_nr1>	<b>NB_NB</b>	<NB_nr2>	The effective notepad number is calculated as: <b>NB_nr_eff</b> = (content from <i>Notepad</i> [NB_nr2]) + NB_nr1 <b>Value</b> = (content from <i>Notepad</i> [NB_nr_eff])

## 5.7 Access to variables - example

Copy the offset value for the third axis (indexed via notepad + offset) of a frame to a notepad:

**AC\_COPY\_DATA (541, P\_C\_FU\_linShift, 49, NB\_ROW, 3, P\_NB, NB\_PA\_308, 0, 0)**

Meaning:

Action routine for copying values (the action ID is 541).

Copy from NC variable *P\_C\_FU\_linShift* for currently displayed channel with row index = content from *Notepad* 49 + 3 (offset) to *Notepad* NB\_PA\_308 (define).

For other examples, please refer to the standard configuring sources.



# 6

## 6 MMC Variables

6.1 MMC variables for OP 030.....	6-212
6.1.1 Variables for alarms and messages .....	6-212
6.1.2 Variables for tables (only OP 030).....	6-214

## 6.1 MMC variables for OP 030

### 6.1.1 Variables for alarms and messages

#### Description

The following MMC variables can be accessed only via output fields (O\_FIELD).

MMC variable	Par1	Par2/Par3	Format
P_ALARM	No meaning	No meaning	CON_STRING
P_DG_ALARM	Number of scrollbar to be used in the alarm/message overview		No meaning

#### P\_Alarm

The number of the most recent alarm is supplied in this variable. If several alarms are active, a downward arrow is displayed on the right of the alarm number. The string (consisting of alarm number and possibly downward arrow) is always 7 characters long.

#### P\_DG\_Alarm

This variable refreshes the internal MMC alarm/message table. Each active alarm/message (maximum character length per alarm: 250), is displayed in a 7-line list and a variable for the cancel criterion for the alarm.

In this list, the first two lines are each 26 characters in length, the following 4 lines are 28 characters in length and the last line contains the remaining 250 characters.

You cannot view alarms directly via this field. They must be displayed via notepads (one notepad for each line of the internal alarm list). The number of the first notepad used is passed in the initialization routine DG\_INIT\_ALARM or DG\_INIT\_MSG.

A field with the variable P\_ALARM must already be active when this field is activated.

#### Example

This field visualizes the first line of the selected 7-line alarm.

```
O_FIELD ( 335, 0, 20, /* ID, x, y */
  30, /* number of characters */
  WHITE, BLACK, /* foreground color, background color*/
  CS_SMALL, /* character set */
  IO_LEFT_ADJUST, /* attribute */
  2, /* refresh cycle */
  P_NB, NB_ALM_VISU, 0, 0, /* MMC variable parameter */
  CON_STRING, 0, 0, 0) /* conversion function/parameter */
```

This field visualizes the last line of the selected 7-line alarm. Since the last line of the alarm is longer than can be displayed on the OP 030 screen, an IO field

is used for this purpose so that you can scroll through the field content (attribute IO\_EDITOR\_SCROLL).

```
IO_FIELD (340, X_IO, Y_IO_1+6*VD,      /* ID, x, y */
          N_IO,      /* width */
          BLACK, WHITE,      /* foreground, background */
          CS_ASIA_SMALL,      /* character-set */
          IO_LEFT_ADJUST |
          IO_EDITOR_SCROLL |
          IO_INPUT_DISABLE,      /* attribute */
          0, 0, 0, 0, 0,      /* dialog cursor */
          2,      /* refresh cycle */
          0,      /* cursor text no */
          P_NB, NB_ALM_VISU + 6, 0, 0, /* xdi_func, xdi parameters */
          CON_STRING, 0, 0, 120)      /* con_func, con parameters */
```

Dummy field for updating the alarm list. This field is not visible externally.

```
O_FIELD ( 345, 0, 0,      /* ID, x, y */
          0,      /* width */
          BLACK, BLACK,      /* foreground, background */
          CS_SMALL,      /* character-set */
          0,      /* attribute */
          2,      /* refresh cycle */
          P_DG_ALARM, SCBAR_IDX, 0, 0, /* xdi_func, xdi parameters */
          CON_HEX, 0, 0, 0)      /* con_func, con parameters */
```

```
/* Scrollbar definition */
/*****
```

```
DEF_SCROLL_BAR (399, /* ID */
                SCBAR_IDX,      /* Nr.Scrollbar: This number
                                must also be specified in the O_FIELD
                                with variable identifier P_DG_ALARM
                                as parameter */
                W_ALARM,      /* window ID */
                X_SB, Y_SB,      /* x,y position */
                WIDTH_SB, HEIGHT_SB, /* width, height */
                WHITE, BLACK,      /* color, background color */
                Y_DIRECTION,      /* direction of motion */
                1,      /* lines per page */
                NB,      /* notepad */
                NB_ALM_CNT,      /* act. No. of alarms */
                NB,
                NB_ALM_PAGE )      /* Index 1st line, actual window */
```

Open list for the window in which the field with variable identifier P\_DG\_ALARM is used. The routine DG\_INIT\_ALARM must be called in this list.

```
BEGIN_OPEN_LIST (OP_W_ALARM)
    AC_DG_INIT_ALARM (210, NB_ALM_VISU,
                    NB_ALM_PAGE, NB_ALM_CNT,
                    NB_ALM_CLEAR_INFO)
END_OPEN_LIST (OP_W_ALARM)
```

### 6.1.2 Variables for tables (only OP 030)

#### Description

The following MMC variables can be accessed only by means of COPY functions (COPY\_DATA, COPY\_DATA\_TO\_NB) in conjunction with tables. In this case, the table must be in this window.

MMC variable	Par1	Par2	Par3
P_GET_ROW	No meaning		
P_GET_COL			
P_PUT_ROW			
P_PUT_COL			

#### P\_GET\_ROW

The row index of the cursor in the table is stored in this variable.

#### Example

In the example below, the row index in the tool overview is copied to a notepad:

```
RC_COPY_DATA (420,          /* COPY_DATA, ID */
              KEY_F5,       /* EVENT, */
              P_GET_ROW,    /* MMC variable, */
              0, 0, 0,      /* Parameter, */
              P_NB,        /* Variable, */
              NB_TV_INDEX, /* notepad number, */
              0, 0)        /* remaining XDI parameters */
```

#### P\_GET\_COL

The column index of the cursor in the table is stored in this variable. In this case, the index configured in element TAB\_COLUMN as a Var server column index is supplied.

**Example**

In the example below, the column index is copied to a notepad:

```
RC_COPY_DATA (420,          /* COPY_DATA, ID */
              KEY_F5,       /* EVENT, */
              P_GET_COL,    /* MMC variable, */
              0, 0, 0,      /* Parameter, */
              P_NB,         /* Variable, */
              NB_COLUMN,    /* notepad number, */
              0, 0)        /* remaining XDI parameters */
```

**P\_PUT\_ROW**

Using this function, you can position the cursor on a particular row within a table. The row index of the cursors is preset.

**Example**

In the example below, the cursor is positioned on the row specified in the notepad.

```
AC_COPY_DATA (530,          /* COPY_DATA, ID */
              P_NB,         /* identifier: from notepad */
              NB_ROW_GOTO,  /* notepad number */
              0, 0,         /* Parameter */
              P_PUT_ROW,    /* MMC variable */
              0, 0, 0)      /* remaining XDI parameters */
```

**P\_PUT\_COL**

Using this function, you can position the cursor in a table on the column configured in element TAB\_COLUMN as the Var server column index. The column index of the cursors is preset.

**Example**

In the example below, the cursor is positioned on the column specified in the notepad.

```
AC_COPY_DATA (530,          /* COPY_DATA, ID */
              P_NB,         /* identifier: Value in notepad */
              NB_COL_GOTO,  /* notepad number */
              0, 0,         /* Parameter */
              P_PUT_COL,    /* MMC variable */
              0, 0, 0)      /* remaining XDI parameters */
```



## Notes



## 7

## 7 Parameter Data Types

<b>A</b>	ac_id .....	<i>UWORD</i>
	acc_class .....	<i>UWORD</i>
	acl_id.....	<i>UWORD</i>
	alm_nr .....	<i>LONG</i>
	attr.....	<i>UWORD</i>
<b>B</b>	bc, fc .....	<i>UWORD</i>
	bit_attr .....	<i>UWORD</i>
	bit_mask.....	<i>UWORD</i>
	blc .....	<i>UWORD</i>
<b>C</b>	cause .....	<i>UWORD</i>
	char_set .....	<i>UWORD</i>
	check.....	<i>UWORD</i>
	cl_info.....	<i>LONG</i>
	cl_mode.....	<i>UWORD</i>
	cll_id.....	<i>UWORD</i>
	cmp_op .....	<i>UWORD</i>
	cmp_val_b.....	<i>WORD</i>
	cmp_val_w .....	<i>WORD</i>
	cmp_val_l.....	<i>LONG</i>
	cmp_val_d.....	<i>DOUBLE</i>
	color .....	<i>UWORD</i>
	column_attr .....	<i>UWORD</i>
	column_index.....	<i>UWORD</i>
	con .....	<i>UWORD</i>
	con_p1 .....	<i>LONG</i>
	con_p2, con_p3.....	<i>UWORD</i>
cur_r, cur_l, cur_d, cur_u .....	<i>UWORD</i>	

	cursor_txt_id.....	ULONG
	cycle.....	WORD
<b>D</b>	d_index.....	UWORD
	d_typ .....	UWORD
	d_vol .....	UWORD
	df_attr.....	UWORD
	df_id, df_id_1, df_id_2.....	UWORD
	dia_id .....	UWORD
	dist .....	UWORD
<b>E</b>	edit_attr.....	UWORD
	ef_id .....	UWORD
	ev_code.....	UWORD
	ev_id .....	UWORD
	evl_id.....	UWORD
<b>F</b>	fc, bc, .....	UWORD
	field_attr .....	ULONG
	fill.....	UWORD
	ftxt_id, txt_id.....	UWORD
<b>G</b>	graphic_list_id .....	UWORD
<b>H</b>	h, w .....	WORD
<b>I</b>	id .....	UWORD
	init_win_id .....	UWORD
	item_attr.....	ULONG
	item_index_typ.....	UWORD
	ix_l, ix_h .....	UWORD
<b>L</b>	le_id .....	UWORD
	lim_check .....	UWORD
	lim_l, lim_h .....	DOUBLE
	line_dist.....	UWORD
	line_index.....	UWORD
	ll_id.....	UWORD

<b>M</b>	mask .....	<i>UWORD</i>
	max_bl_len .....	<i>UWORD</i>
	max_lines .....	<i>UWORD</i>
	menu_id .....	<i>UWORD</i>
	menu_typ .....	<i>UWORD</i>
	modus .....	<i>UWORD</i>
	msg_nr .....	<i>LONG</i>
<b>N</b>	name_nb .....	<i>WORD</i>
	nb_nr, nb_nr_z, nb_nr_q .....	<i>UWORD</i>
	nb_nr_line_index .....	<i>UWORD</i>
	nb_nr_max_lines .....	<i>UWORD</i>
	num_lines .....	<i>UWORD</i>
	nb_tool .....	<i>UWORD</i>
	nb_test .....	<i>UWORD</i>
	nb_index .....	<i>UWORD</i>
	nb_to .....	<i>UWORD</i>
<b>O</b>	obl_id .....	<i>UWORD</i>
	op .....	<i>UWORD</i>
	opl_id .....	<i>UWORD</i>
	opl_mode .....	<i>UWORD</i>
	ovl_id .....	<i>UWORD</i>
<b>P</b>	p_anz .....	<i>UWORD</i>
	par_1, ..., par_n .....	<i>UWORD</i>
	pat_nr .....	<i>UWORD</i>
	pat1 ... pat8 .....	<i>BYTE</i>
	phys_unit .....	<i>UWORD</i>
	pid_makro .....	<i>UWORD</i>
	plim_l, plim_h .....	<i>UWORD</i>
	poly_nr .....	<i>UWORD</i>
	poly_ptr .....	<i>UWORD*</i>
<b>Q</b>	q_index .....	<i>UWORD</i>
	q_vol .....	<i>UWORD</i>
	qd_typ .....	<i>UWORD</i>

<b>R</b>	r_acc_class, w_acc_class .....	<i>UWORD</i>
	rc_id .....	<i>UWORD</i>
	rcl_id .....	<i>UWORD</i>
	refresh .....	<i>UWORD</i>
	row_cnt .....	<i>UWORD</i>
	row_dist .....	<i>UWORD</i>
<b>S</b>	sb_nr	<i>UWORD</i>
	scroll_dir .....	<i>UWORD</i>
	serv_fkt_id .....	<i>UWORD</i>
	serv_fkt_p1 ... serv_fkt_p4 .....	<i>UWORD</i>
	sil_id .....	<i>UWORD</i>
	sk .....	<i>UWORD</i>
	sk_ev_code .....	<i>UWORD</i>
	sk_lines .....	<i>UWORD</i>
	sk_obl_id .....	<i>UWORD</i>
	sk_rcl_id .....	<i>UWORD</i>
	sk_sts .....	<i>UWORD</i>
	sk_txt_nr .....	<i>ULONG</i>
	style .....	<i>UWORD</i>
<b>T</b>	tab_id .....	<i>UWORD</i>
	tab_item_id .....	<i>UWORD</i>
	tab_item_list_id .....	<i>UWORD</i>
	table_col_attr .....	<i>UWORD</i>
	table_col_id .....	<i>UWORD</i>
	table_col_list_id .....	<i>UWORD</i>
	table_id .....	<i>UWORD</i>
	txt_id, ftxt_id .....	<i>UWORD</i>
	txt_id_basis .....	<i>ULONG</i>
	txt_id_l .....	<i>ULONG</i>
	txt_id_menu_name .....	<i>UWORD</i>
	txt_nr .....	<i>LONG</i>
	txt_ptr .....	<i>BYTE*</i>
<b>V</b>	v_adr, v2_adr, v_adr_1, v_adr_h .....	<i>t_btss</i>

	v_item_index .....	t_btss
	v_item_index_p1 ... v_item_index_p3 .....	UWORD
	v_p1 ... v_p4, v2_p1 ... v2_p3, p1_l ... p3_l, p1_h ... p3_h .....	UWORD
	v_txt_1, v_txt_2 .....	BYTE*
	val_attr .....	UWORD
<b>W</b>	w, h .....	WORD
	watch_attr .....	UWORD
	wert_w .....	UWORD
	wert_l .....	LONG
	wert_d .....	DOUBLE
	win_id .....	UWORD
<b>X</b>	x, x1, x2 .....	WORD
<b>Y</b>	y, y1, y2 .....	WORD
<b>Z</b>	zd_typ .....	UWORD



## Notes

# A

## A Appendix

### A.1 References

#### General Documentation

- /BU/** SINUMERIK & SIMODRIVE, Automation Systems for Machine Tools  
Catalog NC 60  
Order No.: E86060-K4460-A101-A9-7600
- /IKPI/** Industrial Communication and Field Devices  
Catalog IK PI  
Order No.: E86060-K6710-A101-B2-7600
- /ST7/** SIMATIC  
Products for Totally Integrated Automation und Micro Automation  
Catalog ST 70  
Order No.: E86060-K4670-A111-A8-7600
- /Z/** MOTION-CONNECT  
Cable, Connectors & System Components for SIMATIC, SINUMERIK,  
MASTERDRIVES and SIMOTION  
Catalog NC Z  
Order No.: E86060-K4490-A001-B1-7600

#### Electronic Documentation

- /CD1/** The SINUMERIK System (11.02 Edition)  
**DOC ON CD**  
(includes all SINUMERIK 840D/840Di/810D/802 and  
SIMODRIVE publications)  
Order No.: 6FC5298-6CA00-0BG3

**User Documentation**

<b>/AUK/</b>	SINUMERIK 840D/810D Short Guide <b>AutoTurn Operation</b> Order No.: 6FC5298-4AA30-0BP2	(09.99 Edition)
<b>/AUP/</b>	SINUMERIK 840D/810D Operator's Guide <b>AutoTurn Graphic Programming System</b> Programming / Setup Order No.: 6FC5298-4AA40-0BP3	(02.02 Edition)
<b>/BA/</b>	SINUMERIK 840D/810D Operator's Guide <b>MMC</b> Order No.: 6FC5298-6AA00-0BP0	(10.00 Edition)
<b>/BAD/</b>	SINUMERIK 840D/840Di/810D Operator's Guide <b>HMI Advanced</b> Order No.: 6FC5298-6AF00-0BP2	(11.02 Edition)
<b>/BAH/</b>	SINUMERIK 840D/840Di/810D Operator's Guide <b>HT 6</b> Order No.: 6FC5298-0AD60-0BP2	(11.02 Edition)
<b>/BAK/</b>	SINUMERIK 840D/840Di/810D <b>Short Guide Operation</b> Order No.: 6FC5298-6AA10-0BP0	(02.01 Edition)
<b>/BAM/</b>	SINUMERIK 810D/840D Operation/Programming <b>ManualTurn</b> Order No.: 6FC5298-6AD00-0BP0	(08.02 Edition)
<b>/BAS/</b>	SINUMERIK 840D/840Di/810D Operation/Programming <b>ShopMill</b> Order No.: 6FC5298-6AD10-0BP2	(11.03 Edition)
<b>/BAT/</b>	SINUMERIK 840D/810D Operation/Programming <b>ShopTurn</b> Order No.: 6FC5298-6AD50-0BP2	(06.03 Edition)
<b>/BEM/</b>	SINUMERIK 840D/810D Operator's Guide <b>HMI Embedded</b> Order No.: 6FC5298-6AC00-0BP2	(11.02 Edition)
<b>/BNM/</b>	SINUMERIK 840D/840Di/810D User's Guide <b>Measuring Cycles</b> Order No.: 6FC5298-6AA70-0BP2	(11.02 Edition)
<b>/BTDI/</b>	SINUMERIK 840D/840Di/810D Motion Control Information System (MCIS) User's Guide <b>Tool Data Information</b> Order No.: 6FC5297-6AE01-0BP0	(04.03 Edition)



<b>/CAD/</b>	SINUMERIK 840D/840Di/810D Operator's Guide <b>CAD-Reader</b> Order No.: (included in online help)	(03.02 Edition)
<b>/DA/</b>	SINUMERIK 840D/840Di/810D <b>Diagnostics Guide</b> Order No.: 6FC5298-6AA20-0BP3	(11.02 Edition)
<b>/KAM/</b>	SINUMERIK 840D/810D Short Guide <b>ManualTurn</b> Order No.: 6FC5298-5AD40-0BP0	(04.01 Edition)
<b>/KAS/</b>	SINUMERIK 840D/810D Short Guide <b>ShopMill</b> Order No.: 6FC5298-5AD30-0BP0	(04.01 Edition)
<b>/KAT/</b>	SINUMERIK 840D/810D Short Guide <b>ShopTurn</b> Order No.: 6FC5298-6AF20-0BP0	(07.01 Edition)
<b>/PG/</b>	SINUMERIK 840D/840Di/810D Programming Guide <b>Fundamentals</b> Order No.: 6FC5298-6AB00-0BP2	(11.02 Edition)
<b>/PGA/</b>	SINUMERIK 840D/840Di/810D Programming Guide <b>Advanced</b> Order No.: 6FC5298-6AB10-0BP2	(11.02 Edition)
<b>/PGK/</b>	SINUMERIK 840D/840Di/810D Short Guide <b>Programming</b> Order No.: 6FC5298-6AB30-0BP0	(10.00 Edition)
<b>/PGM/</b>	SINUMERIK 840D/840Di/810D Programming Guide <b>ISO Milling</b> Order No.: 6FC5298-6AC20-0BP2	(11.02 Edition)
<b>/PGT/</b>	SINUMERIK 840D/840Di/810D Programming Guide <b>ISO Turning</b> Order No.: 6FC5298-6AC10-0BP2	(11.02 Edition)
<b>/PGZ/</b>	SINUMERIK 840D/840Di/810D Programming Guide <b>Cycles</b> Order No.: 6FC5298-6AB40-0BP2	(11.02 Edition)
<b>/PI/</b>	PCIN 4.4 Software for Data Transfer to/from <b>MMC Module</b> Order No.: 6FX2060-4AA00-4XB0 (English, French, German) Order from: WK Fürth	
<b>/SYI/</b>	SINUMERIK 840Di <b>System Overview</b> Order No.: 6FC5298-6AE40-0BP0	(02.01 Edition)

**Manufacturer/Service Documentation****a) Lists  
/LIS/**

SINUMERIK 840D/840Di/810D (11.02 Edition)  
SIMODRIVE 611D  
**Lists**  
Order No.: 6FC5297-6AB70-0BP3

**b) Hardware  
/ASAL/**

SIMODRIVE (10.03 Edition)  
Planning Guide General Information for **Asynchronous Motors**  
Order No.: 6SN1197-0AC62-0BP0

**/APH2/** SIMODRIVE (10.03 Edition)  
Planning Guide **1PH2 Asynchronous Motors**  
Order No.: 6SN1197-0AC63-0BP0

**/APH4/** SIMODRIVE (10.03 Edition)  
Planning Guide **1PH4 Asynchronous Motors**  
Order No.: 6SN1197-0AC64-0BP0

**/APH7/** SIMODRIVE (12.03 Edition)  
Planning Guide **1PH7 Asynchronous Motors**  
Order No.: 6SN1197-0AC65-0BP0

**/APL6/** SIMODRIVE (12.03 Edition)  
Planning Guide **1PL6 Asynchronous Motors**  
Order No.: 6SN1197-0AC66-0BP0

**/BH/** SINUMERIK 840D/840Di/810D (11.02 Edition)  
**Operator Components Manual**  
Order No.: 6FC5297-6AA50-0BP2

**/BHA/** SIMODRIVE Sensor (03.03 Edition)  
User's Guide (HW) **Absolute Position Sensor with Profibus-DP**  
Order No.: 6SN1197-0AB10-0YP2

**/EMV/** SINUMERIK, SIROTEC, SIMODRIVE (06.99 Edition)  
Planning Guide **EMC-Installation Guide**  
Order No.: 6FC5297-0AD30-0BP1

The current Declaration of Conformity is available under the following  
Internet address:

<http://www4.ad.siemens.de>

Please enter the ID No.: 15257461 in the "Search" field (top right) and  
click on "go".

**/GHA/** SINUMERIK/SIMOTION (02.03 Edition)  
**ADI4 – Analog Drive Interface for 4 Axes**  
Manual  
Order No.: 6FC5297-0BA01-0BP1

<b>/PFK6/</b>	SIMODRIVE Planning Guide <b>1FK6 Three-Phase AC Servomotors</b> Order No.: 6SN1197-0AD05-0BP0	(05.03 Edition)
<b>/PFK7/</b>	SIMODRIVE Planning Guide <b>1FK7 Three-Phase AC Servomotors</b> Order No.: 6SN1197-0AD06-0BP0	(01.03 Edition)
<b>/PFS6/</b>	SIMOVERT MASTERDRIVES Planning Guide <b>1FS6 Three-Phase AC Servomotors</b> Order No.: 6SN1197-0AD08-0BP0	(07.03 Edition)
<b>/PFT5/</b>	SIMODRIVE Planning Guide <b>1FT5 Three-Phase AC Servomotors</b> Order No.: 6SN1197-0AD01-0BP0	(05.03 Edition)
<b>/PFT6/</b>	SIMODRIVE 611, SIMOVERT MASTERDRIVES Planning Guide <b>1FT6 Three-Phase AC Servomotors</b> Order No.: 6SN1197-0AD02-0BP0	(12.03 Edition)
<b>/PFU/</b>	SINAMICS, SIMOVERT MASTERDRIVES, MICROMASTER <b>SIEMOSYN Motors 1FU8</b> Order No.: 6SN1197-0AC80-0BP0	(09.03 Edition)
<b>/PHC/</b>	SINUMERIK 810D Configuring Manual <b>CCU (HW)</b> Order No.: 6FC5297-6AD10-0BP1	(11.02 Edition)
<b>/PHD/</b>	SINUMERIK 840D Configuring Manual <b>NCU NC 561.2-573.4 (HW)</b> Order No.: 6FC5297-6AC10-0BP2	(10.02 Edition)
<b>/PJAL/</b>	SIMODRIVE Planning Guide <b>Three-Phase AC Servomotors</b> <b>General Part for 1FT/1FK Motors</b> Order No.: 6SN1197-0AD07-0BP0	(01.03 Edition)
<b>/PJFE/</b>	SIMODRIVE Planning Guide <b>1FE1 Built-In Synchronous Motors</b> Three-Phase AC Motors for Main Spindle Drives Order No.: 6SN1197-0AC00-0BP4	(02.03 Edition)
<b>/PJF1/</b>	SIMODRIVE Installation Guide <b>1FE1 051.-1FE1 147. Built-In Synchronous Motors</b> Three-Phase AC Motors for Main Spindle Drives Order No.: 610.43000.02	(12.02 Edition)
<b>/PJLM/</b>	SIMODRIVE Planning Guide <b>1FN1, 1FN3 Linear Motors</b> ALL General Information about Linear Motors 1FN1 1FN1 Three-Phase AC Linear Motor 1FN3 1FN3 Three-Phase AC Linear Motor CON Connections Order No.: 6SN1197-0AB70-0BP3	(06.02 Edition)

<b>/PJM/</b>	SIMODRIVE Planning Guide <b>Motors</b> Three-Phase AC Servo Motors for Feed and Main Spindle Drives Order No.: 6SN1197-0AA20-0BP4	(11.00 Edition)
<b>/PJM2/</b>	SIMODRIVE Planning Guide <b>Servomotors</b> Three-Phase AC Motors for Feed and Main Spindle Drives Order No.: 6SN1197-0AC20-0BP0	(07.03 Edition)
<b>/PJTM/</b>	SIMODRIVE Planning Guide <b>1FW6 Integrated Torque Motors</b> Order No.: 6SN1197-0AD00-0BP1	(05.03 Edition)
<b>/PJU/</b>	SIMODRIVE 611 Planning Guide <b>Inverters</b> Order No.: 6SN1197-0AA00-0BP6	(02.03 Edition)
<b>/PMH/</b>	SIMODRIVE Sensor Configuring/Installation Guide <b>Hollow-Shaft Measuring System SIMAG H</b> Order No.: 6SN1197-0AB30-0BP1	(07.02 Edition)
<b>/PMHS/</b>	SIMODRIVE Installation Guide <b>Measuring System for Main Spindle Drives</b> <b>SIZAG2 Toothed-Wheel Encoder</b> Order No.: 6SN1197-0AB00-0YP3	(12.00 Edition)
<b>/PMS/</b>	SIMODRIVE Planning Guide <b>ECO Motor Spindle for Main Spindle Drives</b> Order No.: 6SN1197-0AD04-0BP0	(02.03 Edition)
<b>/PPH/</b>	SIMODRIVE Planning Guide <b>1PH2, 1PH4, 1PH7 Motors</b> AC Induction Motors for Main Spindle Drives Order No.: 6SN1197-0AC60-0BP0	(12.01 Edition)
<b>/PPM/</b>	SIMODRIVE Planning Guide Hollow-Shaft Motors for <b>1PM4 and 1PM6</b> Main Spindle Drives Order No.: 6SN1197-0AD03-0BP0	(11.01 Edition)

**c) Software**  
**/FB1/**

SINUMERIK 840D/840Di/810D/FM-NC (09.03 Edition)

Description of Functions **Basic Machine (Part 1)**

(the various sections are listed below)

Order No.: 6FC5297-6AC20-0BP3

A2	Various Interface Signals
A3	Axis Monitoring, Protection Zones
B1	Continuous Path Mode, Exact Stop and Look Ahead
B2	Acceleration
D1	Diagnostic Tools
D2	Interactive Programming
F1	Travel to Fixed Stop
G2	Velocities, Setpoint/Actual-Value Systems, Closed-Loop Control
H2	Output of Auxiliary Functions to PLC
K1	Mode Group, Channel, Program Operation Mode
K2	Axes, Coordinate Systems, Frames, Actual-Value System for Workpiece, External Zero Offset
K4	Communication
N2	EMERGENCY STOP
P1	Transverse Axes
P3	Basic PLC Program
R1	Reference Point Approach
S1	Spindles
V1	Feeds
W1	Tool Offset

**/FB2/**

SINUMERIK 840D/840Di/810D (11.02 Edition)

Description of Functions **Extended Functions (Part 2)**

including FM-NC: Turning, Stepper Motor

(the various sections are listed below)

Order No.: 6FC5297-6AC30-0BP2

A4	Digital and Analog NCK I/Os
B3	Several Operator Panels and NCUs
B4	Operation via PG/PC
F3	Remote Diagnostics
H1	JOG with/without Handwheel
K3	Compensations
K5	Mode Groups, Channels, Axis Replacement
L1	FM-NC Local Bus
M1	Kinematic Transformation
M5	Measurement
N3	Software Cams, Position Switching Signals
N4	Punching and Nibbling
P2	Positioning Axes
P5	Oscillation
R2	Rotary Axes
S3	Synchronous Spindles
S5	Synchronized Actions (up to and including SW 3/higher/FBSY/)
S6	Stepper Motor Control
S7	Memory Configuration
T1	Indexing Axes
W3	Tool Change
W4	Grinding

<b>/FB3/</b>	<p>SINUMERIK 840D/840Di/810D (11.02 Edition)          Description of Functions <b>Special Functions (Part 3)</b>          (the various sections are listed below)          Order No.: 6FC5297-6AC80-0BP2</p> <ul style="list-style-type: none"> <li>F2 3-Axis to 5-Axis Transformation</li> <li>G1 Gantry Axes</li> <li>G3 Cycle Times</li> <li>K6 Contour Tunnel Monitoring</li> <li>M3 Coupled Motion and Leading Value Coupling</li> <li>S8 Constant Workpiece Speed for Centerless Grinding</li> <li>T3 Tangential Control</li> <li>TE0 Installation and Activation of Compile Cycles</li> <li>TE1 Clearance Control</li> <li>TE2 Analog Axis</li> <li>TE3 Master-Slave for Drives</li> <li>TE4 Transformation Package Handling</li> <li>TE5 Setpoint Exchange</li> <li>TE6 MCS Coupling</li> <li>TE7 Retrace Support</li> <li>TE8 Path-Synchronous Switch Signal</li> <li>V2 Preprocessing</li> <li>W5 3D Tool Radius Compensation</li> </ul>	
<b>/FBA/</b>	<p>SIMODRIVE 611D/SINUMERIK 840D/810D (11.02 Edition)          Description of Functions <b>Drive Functions</b>          (the various sections are listed below)          Order No.: 6SN1197-0AA80-1BP0</p> <ul style="list-style-type: none"> <li>DB1 Operation Messages/Alarm Reactions</li> <li>DD1 Diagnostic Functions</li> <li>DD2 Speed Control Loop</li> <li>DE1 Extended Drive Functions</li> <li>DF1 Enable Commands</li> <li>DG1 Encoder Parameterization</li> <li>DL1 Linear Motor MD</li> <li>DM1 Calculation of Motor/Power Section Parameters and Controller Data</li> <li>DS1 Current Control Loop</li> <li>DÜ1 Monitors/Limitations</li> </ul>	
<b>/FBAN/</b>	<p>SINUMERIK 840D/SIMODRIVE 611 digital (02.00 Edition)          Description of Functions <b>ANA MODULE</b>          Order No.: 6SN1197-0AB80-0BP0</p>	
<b>/FBD/</b>	<p>SINUMERIK 840D (07.99 Edition)          Description of Functions <b>Digitizing</b>          Order No.: 6FC5297-4AC50-0BP0</p> <ul style="list-style-type: none"> <li>DI1 Start-up</li> <li>DI2 Scanning with Tactile Sensors (scancad scan)</li> <li>DI3 Scanning with Lasers (scancad laser)</li> <li>DI4 Milling Program Generation (scancad mill)</li> </ul>	
<b>/FBDM/</b>	<p>SINUMERIK 840D/840Di/810D (09.03 Edition)          Description of Functions NC Program Management          DNC Machines          Order No.: 6FC5297-1AE81-0BP0</p>	

<b>/FBDN/</b>	SINUMERIK 840D/840Di/810D Motion Control Information System (MCIS) Description of Functions <b>DNC NC Program Management</b> Order No.: 6FC5297-1AE80-0BP0 DN1 DNC Plant/DNC Cell DN2 DNC IFC SINUMERIK, NC Data Transfer via Network	(03.03 Edition)
<b>/FBFA/</b>	SINUMERIK 840D/840Di/810D Description of Functions <b>ISO Dialects for SINUMERIK</b> Order No.: 6FC5297-6AE10-0BP3	(11.02 Edition)
<b>/FBFE/</b>	SINUMERIK 840D/810D Description of Functions <b>Remote Diagnosis</b> Order No.: 6FC5297-0AF00-0BP2 FE1 Remote Diagnosis ReachOut FE3 Remote Diagnosis pcAnywhere	(04.03 Edition)
<b>/FBH/</b>	SINUMERIK 840D/840Di/810D <b>HMI Configuration Package</b> Order No.: (supplied with the software)  Part 1 User's Guide Part 2 Description of Functions	(11.02 Edition)
<b>/FBH1/</b>	SINUMERIK 840D/840Di/810D <b>HMI Configuration Package</b> <b>ProTool/Pro Option SINUMERIK</b> Order No.: (supplied with the software)	(03.03 Edition)
<b>/FBHL/</b>	SINUMERIK 840D/SIMODRIVE 611 digital Description of Functions <b>HLA Module</b> Order No.: 6SN1197-0AB60-0BP3	(10.03 Edition)
<b>/FBIC/</b>	SINUMERIK 840D/840Di/810D Motion Control Information System (MCIS) Description of Functions <b>TDI Ident Connection</b> Order No.: 6FC5297-1AE60-0BP0	(06.03 Edition)
<b>/FBMA/</b>	SINUMERIK 840D/810D Description of Functions <b>ManualTurn</b> Order No.: 6FC5297-6AD50-0BP0	(08.02 Edition)
<b>/FBO/</b>	SINUMERIK 840D/810D Description of Functions <b>Configuring OP 030 Operator Interface</b> Order No.: 6FC5297-6AC40-0BP0 BA Operator's Guide EU Development Environment (Configuring Package) PSE Introduction to Configuring of Operator Interface IK Screen Kit: Software Update and Configuration	(09.01 Edition)
<b>/FBP/</b>	SINUMERIK 840D Description of Functions <b>C-PLC-Programming</b> Order No.: 6FC5297-3AB60-0BP0	(03.96 Edition)

<b>/FBR/</b>	SINUMERIK 840D/810D IT Solutions Description of Functions <b>Computer Link (SinCOM)</b> Order No.: 6FC5297-6AD60-0BP0 NFL                   Host Computer Interface NPL                   PLC/NCK Interface	(09.01 Edition)
<b>/FBSI/</b>	SINUMERIK 840D/SIMODRIVE Description of Functions SINUMERIK <b>Safety Integrated</b> Order No.: 6FC5297-6AB80-0BP1	(07.02 Edition)
<b>/FBSP/</b>	SINUMERIK 840D/840Di/810D Description of Functions <b>ShopMill</b> Order No.: 6FC5297-6AD80-0BP1	(08.03 Edition)
<b>/FBST/</b>	SIMATIC Description of Functions <b>FM STEPDRIVE/SIMOSTEP</b> Order No.: 6SN1197-0AA70-0YP4	(01.01 Edition)
<b>/FBSY/</b>	SINUMERIK 840D/810D Description of Functions <b>Synchronized Actions</b> Order No.: 6FC5297-6AD40-0BP2	(10.02 Edition)
<b>/FBT/</b>	SINUMERIK 840D/810D Description of Functions <b>ShopTurn</b> Order No.: 6FC5297-6AD70-0BP2	(06.03 Edition)
<b>/FBTC/</b>	SINUMERIK 840D/810D IT Solutions <b>Tool Data Communication SinTDC</b> Description of Functions Order No.: 6FC5297-5AF30-0BP0	(01.02 Edition)
<b>/FBTD/</b>	SINUMERIK 840D/810D IT Solutions <b>Tool Information System (SinTDI)</b> with Online Help Description of Functions Order No.: 6FC5297-6AE00-0BP0	(02.01 Edition)
<b>/FBTP/</b>	SINUMERIK 840D/840Di/810D Motion Control Information System (MCIS) Description of Functions <b>TPM Total Productive Maintenance</b> Order No.: Document is supplied with the software	(01.03 Edition)
<b>/FBU/</b>	SIMODRIVE 611 universal/universal E <b>Closed-Loop Control Component for Speed Control and Positioning</b> Description of Functions Order No.: 6SN1197-0AB20-0BP8	(07.03 Edition)
<b>/FBU2/</b>	SIMODRIVE 611 <b>universal</b> Installation Guide (enclosed with SIMODRIVE 611 universal)	(04.02 Edition)
<b>/FBW/</b>	SINUMERIK 840D/810D Description of Functions <b>Tool Management</b> Order No.: 6FC5297-6AC60-0BP1	(11.02 Edition)



<b>/HBA/</b>	SINUMERIK 840D/840Di/810D Manual <b>@Event</b> Order No.: 6AU1900-0CL20-0BA0	(03.02 Edition)
<b>/HBI/</b>	SINUMERIK 840Di Manual <b>SINUMERIK 840Di</b> Order No.: 6FC5297-6AE60-0BP2	(09.03 Edition)
<b>/INC/</b>	SINUMERIK 840D/840Di/810D System Description <b>Commissioning Tool SINUMERIK SinuCOM NC</b> Order No.: (an integral part of the online help for the start-up tool)	(06.03 Edition)
<b>/PJE/</b>	SINUMERIK 840D/810D Description of Functions <b>Configuring Package HMI Embedded</b> Software Update, Configuration Installation Order No.: 6FC5297-6EA10-0BP0	(08.01 Edition)
<b>/PS/</b>	SINUMERIK 840D/810D <b>Configuring Syntax</b> (The document PS Configuring Syntax is supplied with the software and available as a pdf-file.)	(09.03 Edition)
<b>/POS1/</b>	SIMODRIVE <b>POSMO A</b> User's Guide <b>Distributed Positioning Motor on PROFIBUS DP</b> Order No.: 6SN2197-0AA00-0BP6	(08.03 Edition)
<b>/POS2/</b>	SIMODRIVE <b>POSMO A</b> Installation Guide (enclosed with POSMO A)	(05.03 Edition)
<b>/POS3/</b>	SIMODRIVE POSMO SI/CD/CA User's Guide <b>Distributed Servo Drive Systems</b> Order No.: 6SN2197-0AA20-0BP5	(07.03 Edition)
<b>/POS4/</b>	SIMODRIVE <b>POSMO SI</b> Installation Guide (enclosed with POSMO SI)	(04.02 Edition)
<b>/POS5/</b>	SIMODRIVE <b>POSMO CD/CA</b> Installation Guide (enclosed with POSMO CD/CA)	(04.02 Edition)
<b>/S7H/</b>	SIMATIC S7-300 Installation Manual <b>Technological Functions</b> Order No.: 6ES7398-8AA03-8BA0 - Reference Manual: CPU Data (Hardware) - Reference Manual: Module Data	(2002 Edition)
<b>/S7HT/</b>	SIMATIC S7-300 Manual: <b>STEP 7, Fundamentals, V. 3.1</b> Order No.: 6ES7810-4CA02-8BA0	(03.97 Edition)
<b>/S7HR/</b>	SIMATIC S7-300 Manual: <b>STEP 7, Reference Manuals, V. 3.1</b> Order No.: 6ES7810-4CA02-8BR0	(03.97 Edition)

<b>/S7S/</b>	SIMATIC S7-300 <b>FM 353 Positioning Module for Stepper Drive</b> Order together with configuring package	(04.02 Edition)
<b>/S7L/</b>	SIMATIC S7-300 <b>FM 354 Positioning Module for Servo Drive</b> Order together with configuring package	(04.02 Edition)
<b>/S7M/</b>	SIMATIC S7-300 <b>FM 357.2 Multimodule</b> for Servo and Stepper Drives Order together with configuring package	(01.03 Edition)
<b>/SP/</b>	SIMODRIVE 611-A/611-D <b>SimoPro 3.1</b> Program for Configuring Machine-Tool Drives Order No.: 6SC6111-6PC00-0BA□ Order from: WK Fürth	
<b>d) Installation and Start-up</b>		
<b>/BS/</b>	SIMODRIVE 611 analog Description <b>Start-Up Software for Main Spindle and Asynchronous Motor Modules Version 3.20</b> Order No.: 6SN1197-0AA30-0BP1	(10.00 Edition)
<b>/IAA/</b>	SIMODRIVE 611A <b>Installation and Start-Up Guide</b> Order No.: 6SN1197-0AA60-0BP6	(10.00 Edition)
<b>/IAC/</b>	SINUMERIK 810D <b>Installation and Start-Up Guide</b> (incl. description of SIMODRIVE 611D start-up software) Order No.: 6FC5297-6AD20-0BP1	(11.02 Edition)
<b>/IAD/</b>	SINUMERIK 840D/SIMODRIVE 611D <b>Installation and Start-Up Guide</b> (incl. description of SIMODRIVE 611D digital start-up software SIMODRIVE 611D) Order No.: 6FC5297-6AB10-0BP2	(11.02 Edition)
<b>/IAM/</b>	SINUMERIK 840D/840Di/810D <b>Installation and Start-Up Guide HMI/MMC</b> Order No.: 6FC5297-6AE20-0BP2 AE1 Updates/Supplements BE1 Expanding the Operator Interface HE1 Online Help IM2 Starting up HMI Embedded IM4 Starting up HMI Advanced TX1 Creating Foreign Language Texts	(11.02 Edition)



# I Index

## I.1 Keyword index

### A

Action element .....	2-33
Action field .....	3-116
Action list .....	2-33
<b>ACTIVATE_SK_GRAPHIC</b> .....	4-156
Address structure .....	5-198
Alarm .....	6-214
<b>Alarm overview</b> .....	4-190
<b>APPEND_TXT_NB_TXT</b> .....	4-170
<b>APPEND_TXT_NB_TXT_NB</b> .....	4-171
ARC .....	3-62
Arc, sector .....	3-62
ARC_DYN .....	3-63
Area .....	5-198
ARROW .....	3-58
ARROW_DYN .....	3-58
Arrowhead .....	3-58

### B

BIT_EVENT .....	2-35
Bitmaps .....	3-123
Block type .....	5-199
<b>BREAK_EVENT</b> .....	4-173
<b>BREAK_IF</b> .....	2-45
<b>BREAK_UNCOND</b> .....	2-44
<b>BV_LANGUAGE_CHANGE</b> .....	4-183

### C

<b>CALC_DATA</b> .....	4-178
<b>CALC_DOUBLE</b> .....	4-177
<b>CALC_LONG</b> .....	4-177
<b>CALC_UWORD</b> .....	4-177
<b>CHANGE_MODE</b> .....	4-182
<b>CHANNEL_SWITCH</b> .....	4-182
CHECK_FIELD .....	3-94
Circle .....	3-60
CIRCLE .....	3-60
CIRCLE_DYN .....	3-61
<b>CLEAR_TXT_NB</b> .....	4-171
Close list .....	2-35

<b>CLOSE_VERSION</b> .....	4-193
COL_TAB .....	3-66
Color palett .....	3-66
Column .....	5-198
Column index .....	5-200
CON_ASCII .....	3-131
CON_BCD .....	3-136
CON_BINARY .....	3-136
CON_BIT .....	3-137
CON_DECIMAL .....	3-134
CON_HEX .....	3-135
CON_OFF .....	3-137
CON_STRING .....	3-132
CON_STRING_LIMIT .....	3-133
CON_TEXT .....	3-129
CON_TEXT_BOOL .....	3-131
CON_TEXT_OFFSET .....	3-130
Configuring syntax .....	1-16
<b>CONVERT_DATA_FORMAT</b> .....	4-181
<b>COPY_BÖÖCK_NCK_NB</b> .....	4-180
<b>COPY_DATA</b> .....	4-179
<b>COPY_DATA_TO_NB</b> .....	4-179
<b>Create tool</b> .....	4-186
CUR_PICT_FIELD .....	3-93

### D

<b>D_ACTVATE_ACTION</b> .....	4-167
Data conversion .....	3-129
Data refresh .....	3-127
DEF_PATTERN .....	3-65
DEF_POLYMARKER .....	3-72
<b>Deleting a tool</b> .....	4-187
Development kit .....	1-16
Development Kit .....	1-16
<b>DG_CHG_PASSW</b> .....	4-195
<b>DG_CLOSE_PASSW</b> .....	4-194
<b>DG_INIT_ALARM</b> .....	4-190, 6-214
<b>DG_INIT_MSG</b> .....	4-191
<b>DG_INIT_PASSW</b> .....	4-193
<b>DG_SET_PASSW</b> .....	4-194
<b>Dialog fields</b> .....	3-55
Display refresh .....	3-127
Documentation .....	1-16
Dynamic arrowhead .....	3-58

Dynamic circle.....	3-61
Dynamic circle, sector.....	3-63
Dynamic display element.....	3-83
Dynamic ellipse.....	3-64
<b>dynamic line</b> .....	3-57
Dynamic macro element.....	3-121
Dynamic pixel.....	3-56
Dynamic polymarker.....	3-78
Dynamic rectangle.....	3-59
Dynamic text.....	3-70

**E**

Edit field.....	3-97
EDIT_FIELD.....	3-97
EDIT_FIELD_32.....	3-100
Ellipse.....	3-64
ELLIPSE.....	3-64
ELLIPSE_DYN.....	3-64
ENABLE_SK_VISUALISATION.....	4-156
Event list.....	2-35
Exteranal list reference.....	2-25

**F**

fill pattern.....	3-65
FIXTEXT.....	3-66

**G**

<b>GOTO_LABEL</b> .....	2-51
Graphic field list.....	3-93

**H**

<b>Help symbol</b> .....	4-185
--------------------------	-------

**I**

I_FIELD.....	3-90
Initialization list.....	2-42
Input field.....	3-90
Input limit value list.....	2-42
Input/output field.....	3-83
Installation.....	1-16
Intelligent cursor control.....	4-162
INVERSE_FIELD.....	3-118
IO_FIELD.....	3-83

**K**

Keyword input.....	3-84
--------------------	------

**L**

<b>LABEL</b> .....	2-50
Line.....	3-57
LINE.....	3-57
Line index.....	5-200
<b>LINE_DYN</b> .....	3-57
List directory.....	2-24
List identity.....	2-25
List pointer.....	2-25
List type.....	2-25

**M**

MACRO.....	3-120
Macro element.....	3-120
MACRO_DYN.....	3-121
Menu definition block.....	2-24
Menu, global.....	2-27
Menu, local.....	2-27
Message overview.....	4-191
MMC variables.....	1-17

**N**

<b>NB_DECREMENT</b> .....	4-168
NCK interface.....	1-17
NCK version display.....	4-192
new_menu.....	4-142
Notepad entry.....	4-168
Number of rows.....	5-200

**O**

OB_DO_ACTION_LIST.....	2-51
Object list.....	2-31, 3-55
Open list.....	2-34
<b>OPEN_VERSION</b> .....	4-192
Operator's Guide.....	1-16
OPI.....	5-201
Option boxes.....	3-94
Output field.....	3-86

**P**

<b>P_Alarm</b> .....	6-212
<b>P_GET_COL</b> .....	6-214
<b>P_GET_ROW</b> .....	6-214
<b>P_PUT_COL</b> .....	6-215
<b>P_PUT_ROW</b> .....	6-215
Password dialog.....	4-193
PCX-Format.....	3-123
<b>PG_DG_Alarm</b> .....	6-212
PICT_FIELD.....	3-114
Pixel.....	3-56
<b>PIXEL</b> .....	3-56

- PIXEL\_DYN ..... 3-56  
 Polymarker ..... 3-78  
 Polymarker definition ..... 3-72  
 POLYMARKER\_DYN ..... 3-78  
**PP\_EDIT\_CLOSE** ..... 4-189  
**PP\_EDIT\_OPEN** ..... 4-188  
**PP\_REFRESH** ..... 4-189  
 Processing the action list ..... 4-167  
 Progress bar ..... 3-121  
 PROGRESS\_BAR ..... 3-88, 3-121
- R**
- Reaction element ..... 2-39  
 Reaction list ..... 2-39  
 Rectangle ..... 3-59  
 RECTANGLE ..... 3-59  
 RECTANGLE\_DYN ..... 3-59  
 Refresh ..... 4-167  
 Requirements ..... 1-16  
**RESET\_INFO** ..... 4-185  
**RESET\_MESSAGE** ..... 4-183  
**RESET\_MORE** ..... 4-157  
**RESET\_RECALL** ..... 4-157  
 Reverse video field ..... 3-118  
 Row ..... 5-198
- S**
- SCROLL\_BAR\_REFRESH** ..... 4-172  
**Scrollbar** ..... 4-172, 6-213  
 Scrollbar ..... 3-118  
 Search for tool ..... 4-186  
**SET\_BYTE** ..... 4-176  
**SET\_CUR\_TXT\_POS** ..... 4-185  
**SET\_DOUBLE** ..... 4-176  
**SET\_EVENT\_REACTION** ..... 4-173  
**SET\_INFO** ..... 4-185  
**SET\_LONG** ..... 4-176  
**SET\_MESSAGE** ..... 4-183  
**SET\_MORE** ..... 4-157  
**SET\_MSG\_POS** ..... 4-184  
**SET\_RECALL** ..... 4-157  
**SET\_TXT\_NB** ..... 4-169  
**SET\_WORD** ..... 4-176  
**SKIP\_IF** ..... 2-47  
**SKIP\_UNCOND** ..... 2-44  
 soft key object list ..... 2-32  
**Soft key object list** ..... 3-55  
 soft key reaction list ..... 2-40  
 SOFTKEY ..... 3-79
- Standard operating area ..... 2-24  
 Standard symbols ..... 3-69  
 Sub-area ..... 5-205  
**Syntax ID** ..... 5-198  
 System initialization list ..... 2-24
- T**
- TAB\_COLUMN ..... 3-103  
 TAB\_ITEM ..... 3-106  
 Table ..... 3-102  
 TABLE ..... 3-102  
 Table column ..... 3-103  
 TACHO ..... 3-122  
 Tacho element ..... 3-122  
 TACHOMETER ..... 3-89  
 TEXT ..... 3-66  
**Text variable CLEAR\_TXT\_NB** ..... 4-171  
 TEXT\_DYN ..... 3-70  
**TOOL\_CREATE** ..... 4-186  
**TOOL\_DELETE** ..... 4-187  
**TOOL\_SEARCH** ..... 4-186
- U**
- Unit ..... 5-199  
 User operating area ..... 2-24
- V**
- VALUE\_EVENT ..... 2-35  
**Variable MMC** ..... 5-209  
 Variable MMC - Configuring ..... 5-210  
 Variable NCK ..... 5-198  
 Variable NCK - configuring ..... 5-201  
 Variable PLC ..... 5-204, 5-206  
 Variable types ..... 5-198  
 Variables ..... 1-17  
 Version 06.02 ..... 3-123
- W**
- WATCH\_EVENT ..... 2-35  
 Window definition ..... 2-28  
 Window definition block ..... 2-24





To  
Siemens AG

A&D MC BMS  
P.O. Box 3180

D-91050 Erlangen, Germany

Phone ++49-180-5050-222 [Hotline]

Fax ++49-09131-98-2176

E-mail [motioncontrol.docu@erf.siemens.de](mailto:motioncontrol.docu@erf.siemens.de))

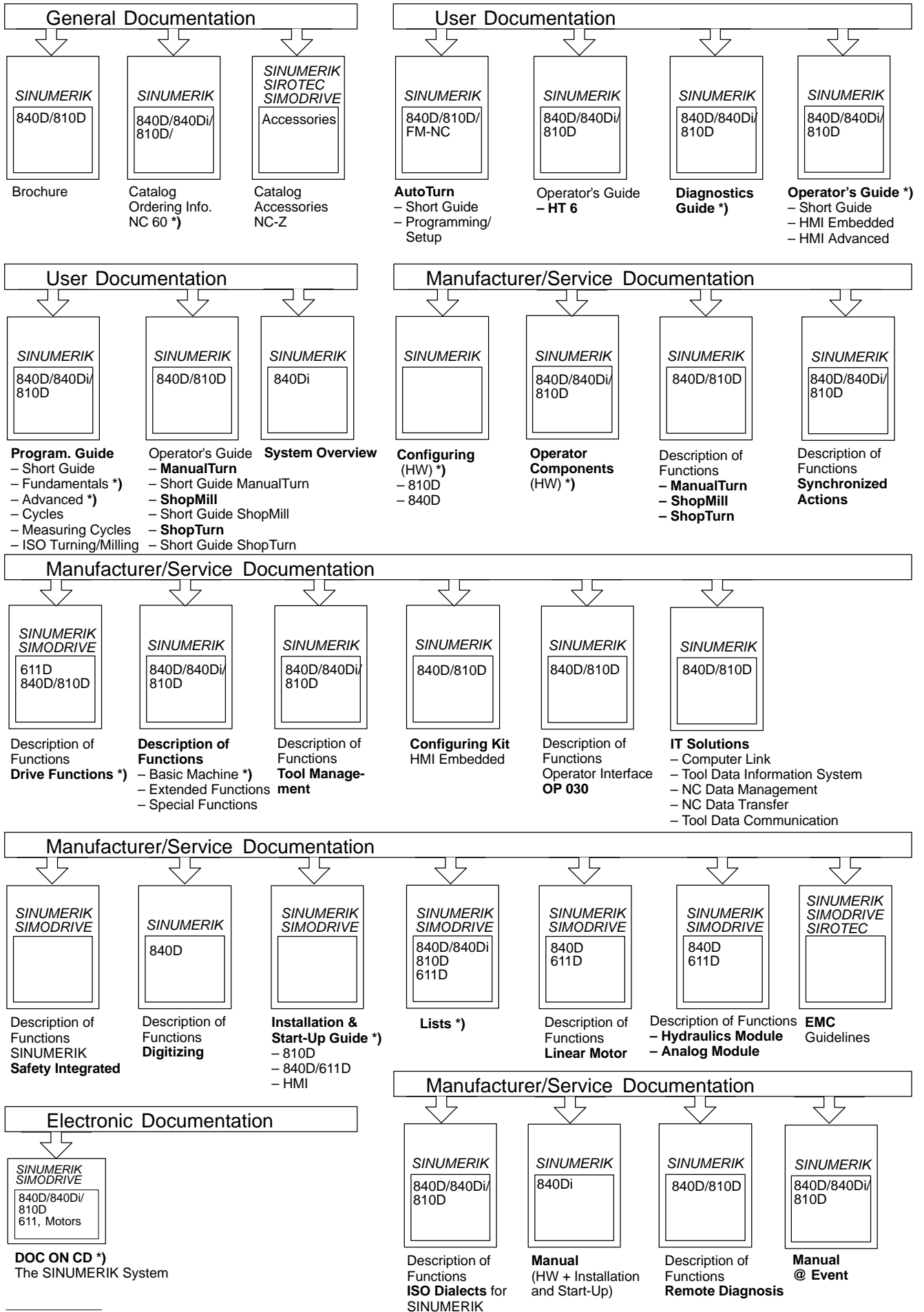
<b>From</b> Name: _____  Company/Dept. _____ Address: _____ Zip code: _____ City: _____ Phone: _____ / _____ Fax: _____ / _____	<b>Suggestions</b> <b>Corrections</b> For Publication/Manual:  SINUMERIK 840D/810D Configuring Syntax Manufacturer Documentation  Planning Guide Order No.:        Only Online 09.03 Edition  Should you come across any printing errors when reading this publication, please notify us using this form. We would also be grateful for ideas and suggestions on how we can improve this documentation.
--	---

**Suggestions and/or corrections**

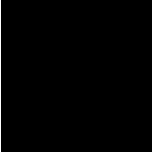




# Overview of SINUMERIK 840D/840Di/810D Documentation



\*) These documents are a minimum requirement



**Siemens AG**

Automation and Drives

Motion Control Systems

P. O. Box 3180, D – 91050 Erlangen

Germany

[www.siemens.com](http://www.siemens.com)

© Siemens AG 2003  
Subject to change without prior notice  
Order No.: Only Online