

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>7</b>
1.1	About NORD CON .....	7
1.2	How to use NORD CON.....	7
<b>2</b>	<b>Graphic user interface .....</b>	<b>10</b>
2.1	Structure of the program interface .....	10
2.2	Main menu .....	11
2.2.1	Category "File".....	11
2.2.2	Category "Edit" .....	12
2.2.3	Category "Project" .....	13
2.2.4	Category "Device" .....	13
2.2.5	Category "View".....	15
2.2.6	Category "Extras" .....	16
2.2.7	Category "Help" .....	17
2.3	Toolbars.....	17
2.3.1	Standard.....	17
2.3.2	Device .....	18
2.3.3	Start.....	19
2.4	View "Project".....	19
2.4.1	Structure of context menu .....	20
2.5	View "Messages" .....	22
2.6	View "Remote" .....	23
2.7	Docking and Undocking .....	24
<b>3</b>	<b>Communication .....</b>	<b>30</b>
3.1	USS.....	30
3.1.1	General settings .....	30
3.1.2	Bus scan.....	31
3.2	Ethernet .....	32
3.2.1	General settings .....	33
3.2.2	Scan .....	34
<b>4</b>	<b>Parameterization .....</b>	<b>36</b>
4.1	How to manipulate parameters .....	36
4.2	Selective parameterisation.....	37
4.3	Off-line Parameterisation .....	37
4.4	Parameter Viewing.....	38
4.5	Comparison report .....	38
4.6	Parameter upload from device .....	39
4.7	Parameter download to device.....	40
<b>5</b>	<b>Control .....</b>	<b>42</b>
5.1	Overview .....	42
5.2	Standard control.....	43
5.3	Detailed control .....	43
5.3.1	Overview .....	43
5.3.2	Control.....	44
5.3.3	Management of setting values and actual values.....	45
5.3.4	Formatting of Setpoint and/or actual value .....	46
5.3.5	Status word.....	46
5.3.6	Control word .....	47
<b>6</b>	<b>Remote.....</b>	<b>49</b>
6.1	Standard .....	49
6.2	NORDAC SK 200 E .....	50
6.3	NORDAC SK 700/500/300 E .....	51
6.4	NORDAC vector mc.....	53
6.5	NORDAC vector ct.....	54
<b>7</b>	<b>Oscilloscope.....</b>	<b>56</b>

7.1	Overview .....	56
7.2	Display .....	56
7.3	Handling.....	57
7.4	Measurement .....	59
7.5	Save and Print .....	60
<b>8</b>	<b>Macro editor.....</b>	<b>61</b>
8.1	User interfaces and views .....	61
8.1.1	Window "Variables" .....	61
8.1.2	Properties window .....	61
8.1.3	Log window.....	64
8.2	Working with macros.....	64
8.2.1	Create a new macro .....	64
8.2.2	Open a macro.....	64
8.2.3	Save a macro .....	65
8.2.4	Inserting instructions.....	65
8.2.5	Copying instructions .....	65
8.2.6	Cutting instructions .....	65
8.2.7	Delete from instruction.....	65
8.2.8	Search and replace .....	65
8.2.9	Shift up instruction.....	65
8.2.10	Shift down instruction .....	65
8.2.11	Creating new instructions .....	66
8.3	Scheduler.....	67
8.3.1	Start sequence .....	67
8.3.2	Cancel a macro .....	67
8.3.3	Execute next instruction .....	67
<b>9</b>	<b>USS Frame-Editor.....</b>	<b>68</b>
9.1	Master (order) .....	70
9.2	Device (response).....	70
<b>10</b>	<b>PLC.....</b>	<b>72</b>
10.1	General .....	72
10.1.1	Specification of the PLC .....	72
10.1.2	PLC structure.....	73
10.1.2.1	Memory .....	73
10.1.2.2	Image of the process .....	73
10.1.2.3	Program Task .....	74
10.1.2.4	Setpoint processing .....	74
10.1.2.5	Data processing via accumulator .....	74
10.1.3	Scope of functions .....	75
10.1.3.1	Motion Control Lib .....	75
10.1.3.2	Electronic gear with Flying Saw .....	75
10.1.3.3	Visualisation .....	75
10.1.3.4	Process controller .....	75
10.1.3.5	CANopen communication .....	76
10.2	Creation of PLC programs .....	76
10.2.1	Loading, saving and printing.....	76
10.2.2	Editor .....	76
10.2.2.1	Variables and FB declaration .....	77
10.2.2.2	Input window .....	78
10.2.2.3	Watch and Breakpoint display window .....	78
10.2.2.4	PLC message window .....	79
10.2.3	Transfer PLC program to device.....	79
10.2.4	Debugging .....	80
10.2.4.1	Observation points (Watchpoints) .....	80
10.2.4.2	Holding points (Breakpoints) .....	80
10.2.4.3	Single Step .....	80
10.2.5	PLC configuration .....	81
10.3	Function blocks .....	81
10.3.1	CANopen.....	81
10.3.1.1	Overview .....	82
10.3.1.2	FB_NMT .....	82
10.3.1.3	FB_PDOConfig .....	83
10.3.1.4	FB_PDOReceive .....	85
10.3.1.5	FB_PDOSend .....	87

10.3.2	Electronic gear unit with flying saw.....	89
10.3.2.1	Overview	90
10.3.2.2	FB_FlyingSaw	90
10.3.2.3	FB_Gearing	91
10.3.3	Motion Control.....	92
10.3.3.1	MC_Control	93
10.3.3.2	MC_Control_MS	94
10.3.3.3	MC_Home	96
10.3.3.4	MC_MoveAbsolute	97
10.3.3.5	MC_MoveAdditive	98
10.3.3.6	MC_MoveRelative	99
10.3.3.7	MC_MoveVelocity	100
10.3.3.8	MC_Power	101
10.3.3.9	MC_ReadActualPos	102
10.3.3.10	MC_ReadParameter	102
10.3.3.11	MC_ReadStatus	103
10.3.3.12	MC_Reset	104
10.3.3.13	MC_Stop	105
10.3.3.14	MC_WriteParameter_16 / MC_WriteParameter_32	105
10.3.4	Standard.....	106
10.3.4.1	CTD downward counter	106
10.3.4.2	CTU upward counter	107
10.3.4.3	CTUD upward and downward counter	108
10.3.4.4	R_TRIG und F_TRIG	110
10.3.4.5	RS Flip Flop	111
10.3.4.6	SR Flip Flop	111
10.3.4.7	TOF switch-off delay	112
10.3.4.8	TON switch-on delay	113
10.3.4.9	TP time pulse	114
10.3.5	Access to memory areas of the frequency inverter.....	115
10.3.5.1	FB_ReadTrace	115
10.3.5.2	FB_WriteTrace	116
10.3.6	Visualisation with ParameterBox.....	117
10.3.6.1	Overview visualisation	118
10.3.6.2	FB_DINTToPBOX	118
10.3.6.3	FB_STRINGToPBOX	121
10.3.7	FB_Capture (Detection of rapid events).....	123
10.3.8	FB_DinCounter.....	125
10.3.9	FB_FunctionCurve.....	126
10.3.10	FB_PIDT1.....	127
10.3.11	FB_ResetPostion.....	129
10.3.12	FB_Weigh.....	130
10.4	Operators.....	131
10.4.1	Arithmetical operators.....	131
10.4.1.1	ABS	131
10.4.1.2	ADD and ADD(	132
10.4.1.3	DIV and DIV(	132
10.4.1.4	LIMIT	133
10.4.1.5	MAX	133
10.4.1.6	MIN	133
10.4.1.7	MOD and MOD(	134
10.4.1.8	MUL and MUL(	134
10.4.1.9	MUX	135
10.4.1.10	SUB and SUB(	135
10.4.2	Extended mathematical operators.....	136
10.4.2.1	COS, ACOS, SIN, ASIN, TAN, ATAN	136
10.4.2.2	EXP	137
10.4.2.3	LN	137
10.4.2.4	LOG	138
10.4.2.5	SQRT	138
10.4.3	Bit operators.....	139
10.4.3.1	AND and AND(	139
10.4.3.2	ANDN and ANDN(	139
10.4.3.3	NOT	140
10.4.3.4	OR and OR(	140
10.4.3.5	ORN and ORN(	141
10.4.3.6	ROL	142
10.4.3.7	ROR	142

10.4.3.8 S and R	143
10.4.3.9 SHL	143
10.4.3.10 SHR	143
10.4.3.11 XOR and XOR(	144
10.4.3.12 XORN and XORN(	144
10.4.4 Loading and storage operators (AWL).....	145
10.4.4.1 LD	145
10.4.4.2 LDN	146
10.4.4.3 ST	146
10.4.4.4 STN	146
10.4.5 Comparison operators .....	146
10.4.5.1 EQ	147
10.4.5.2 GE	147
10.4.5.3 GT	147
10.4.5.4 LE	148
10.4.5.5 LT	148
10.4.5.6 NE	149
10.5 Processing values .....	149
10.5.1 Inputs and outputs .....	150
10.5.2 PLC setpoints and actual values .....	155
10.5.3 Bus setpoints and actual values .....	158
10.5.4 ControlBox and ParameterBox .....	160
10.5.5 Info parameters .....	161
10.5.6 PLC errors .....	164
10.5.7 PLC parameters .....	164
10.6 Languages .....	166
10.6.1 Instruction list (AWL / IL) .....	166
10.6.1.1 General	166
10.6.2 Structured text (ST) .....	169
10.6.2.1 Common	169
10.6.2.2 Procedure	171
10.7 Jumps .....	173
10.7.1 JMP .....	173
10.7.2 JMPC .....	174
10.7.3 JMPCN .....	174
10.8 Type conversion.....	174
10.8.1 BOOL_TO_BYTE .....	174
10.8.2 BYTE_TO_BOOL .....	175
10.8.3 BYTE_TO_INT .....	175
10.8.4 DINT_TO_INT .....	176
10.8.5 INT_TO_BYTE .....	176
10.8.6 INT_TO_DINT .....	177
10.9 PLC Error messages.....	177
<b>11 Project Mode.....</b>	<b>179</b>
11.1 General .....	179
11.2 HMI .....	180
11.3 Save and restore.....	180
<b>12 Firmware .....</b>	<b>183</b>
12.1 How to update the firmware .....	183
12.2 Firmware update program.....	185
12.3 Firmware update via system bus.....	188
<b>13 Settings .....</b>	<b>191</b>
13.1 User interface.....	191
13.2 Device report.....	192
13.3 Control .....	193
13.4 Project.....	194
13.5 Directories.....	195
13.6 Macro editor .....	196
13.7 Parameter .....	197
13.8 PLC.....	197
<b>14 Messages .....</b>	<b>198</b>
14.1 Errors and informations.....	198

<b>15</b>	<b>NORD DRIVESYSTEMS.....</b>	<b>203</b>
15.1	NORD DRIVESYSTEMS corporate history.....	204
15.2	Frequency Inverters .....	206
15.2.1	SK 135E .....	206
15.2.2	SK 180E .....	207
15.2.3	SK 200E .....	207
15.2.4	SK 500E .....	209



## 1 Introduction

### 1.1 About NORD CON

NORD CON is a PC program intended to 4 "Parameterization" and 5.1 "Overview" the inverters and option modules produced by Getriebebau NORD.

With NORD CON, up to 31 frequency inverters can be controlled simultaneously via the integrated RS485 interface. Communication with the frequency inverters is handled by the PC's serial interface.

To enable trial runs or system start-ups, the connected frequency inverters can be controlled via the PC. The program also provides for continuous monitoring of the current status of the frequency inverter while these activities are going on. Complete process sequences can be developed using macros.

With NORD CON, you can perform, document, and save the parameter settings of a frequency inverter which will be read out by the inverter or transmitted to it respectively. Parameter databases can be created or manipulated off-line - i.e. without a frequency inverter being connected.

The program further provides for remote control of the connected frequency inverters. For the frequency inverter to be remote-controlled the operating unit of the type in question is simulated on the PC. This is a convenient way of operating devices which are either difficult to access or haven't got an operating unit themselves.

8 "Macro editor"

6 "Remote"

### 1.2 How to use NORD CON

---

#### Information

#### Serial interface

For the parameterisation and controlling of the devices with NORD CON, your PC requires a serial interface.

---

##### 1. Installation

Please start the installation program of NORD CON on the enclosed CD or load the installation program from the Internet ("<http://www2.nord.com/cms/de/documentation/software/software-overview.jsp>"). Enter all necessary information and install NORD CON into the standard directory.

##### 2. Connect

If the frequency inverter is equipped with an RS232 optional interface, it can be directly connected to the PC with a serial 1-1 cable. In this case, only one frequency inverter can be connected. Each NORD Frequency inverter frequency inverter features an integrated RS485 interface which can be activated via the control terminals. This interface allows for configuration of a master/slave bus system with up to 31 devices max. For NORD CON to be connected to such a bus, an RS232 - RS485 converter will be required.

**i Information****USS Settings**

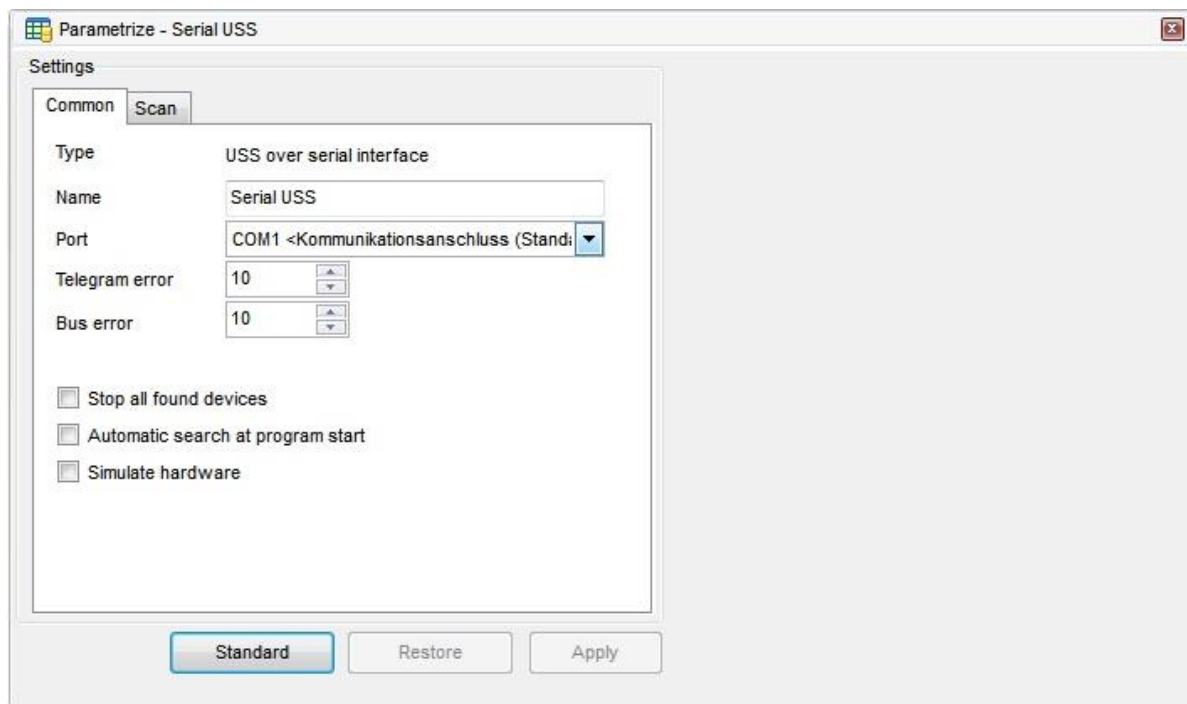
If several devices are operated simultaneously, make absolutely sure that a unique USS address is assigned to each of the devices connected, and that all of them have the same 3.1.2 "Bus scan" setting (see also Operating Instructions of the frequency inverter type involved).

**3. Run NORD CON**

In order to start NORD CON, you use the shortcut "NORD CON start" or "Start->Program->->Nord->NORD CON 2.5->NORD CON".

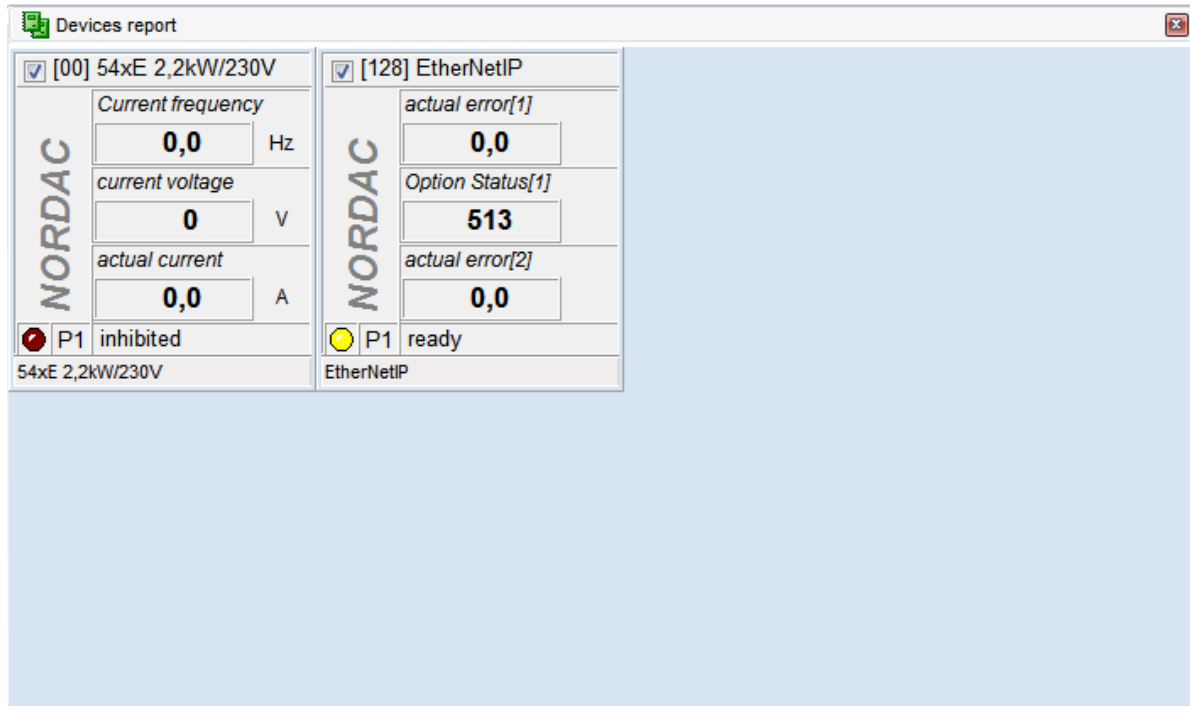
**4. Setup of the communication module**

In order to set the communication parameters, one must select the appropriate module in the project view. Over the menu entry "Device-> Parameterise" the parameter dialogue of the module can be opened. In the edit field "Port" must be inserted the correct COM port number. After that you have to push the button "Apply". Additional settings are not necessary for the first application and the window can be closed.

**5. Bus scan**

After the start of bus scan, all ready and connected devices are searched for. All found devices are represented in the project tree and in the equipment overview. Subsequently, the first device in the list is marked and the users can use all device-specific functions.





## 6. Work with the devices

The user can now select the device by clicking the device in the device overview or in the project tree. Functions, like control or parametrises, are available in the popup menu of the project tree, the tool bar or the main menu.

2.4 "View "Project""

2.3 "Toolbars"

5 "Control"

4 "Parameterization"

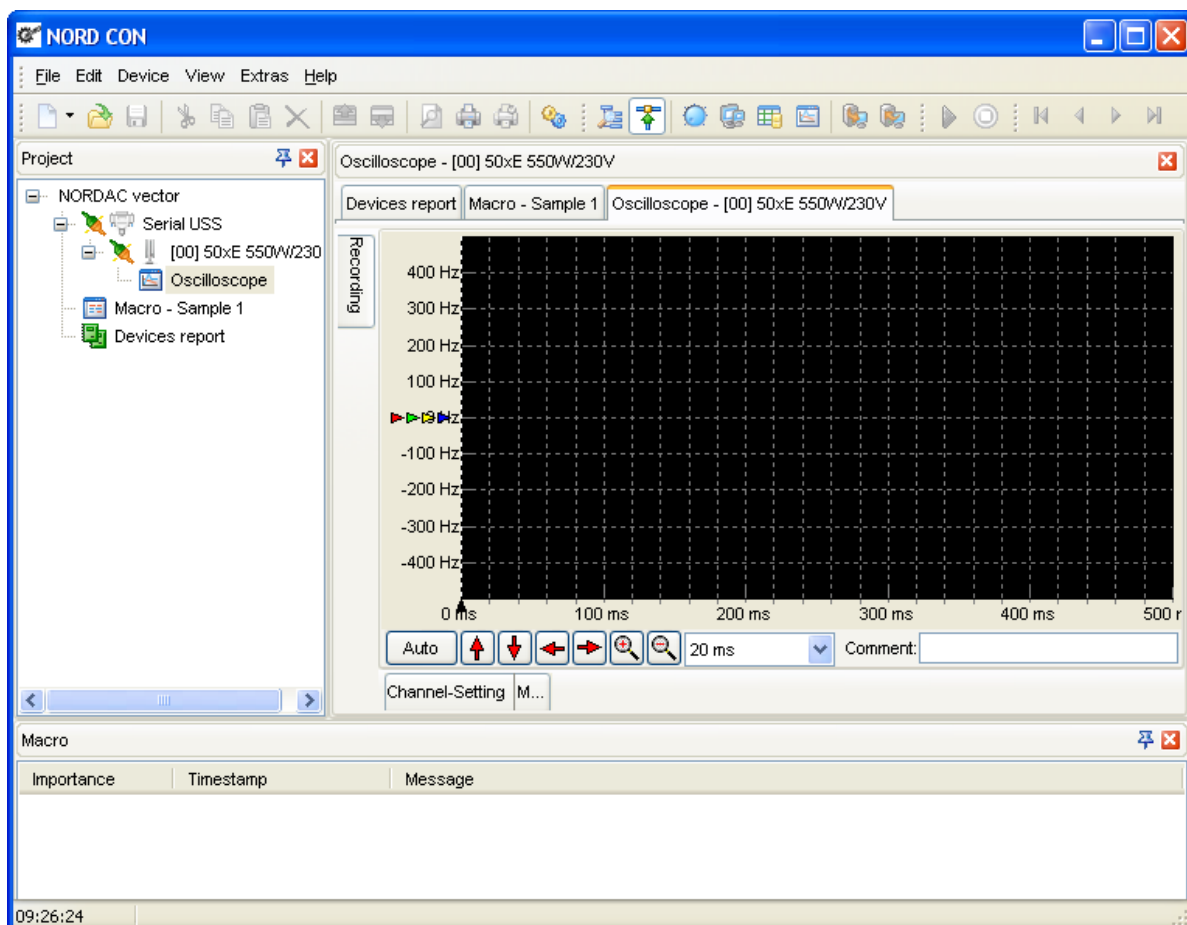
## 2 Graphic user interface

### 2.1 Structure of the program interface

The main window consists of main menu, toolbars, work area, and the different views. In the work area the different editor windows like parameter windows or macros are displayed. The windows can be positioned freely or can be docked at the sides of the work area. In order to change the position of a docked window, click on the header bar of the window and keep the mouse button pressed. Subsequently, the new position can be specified with the pointer of mouse. A coloured rectangle shows the current position and dock condition. After releasing the left mouse button, the actual action is implemented. In addition, the user can dock or undock the window by clicking on the header bar. The layout is stored when closing application and resumed with the restart.

**The interface is divided into the following areas:**

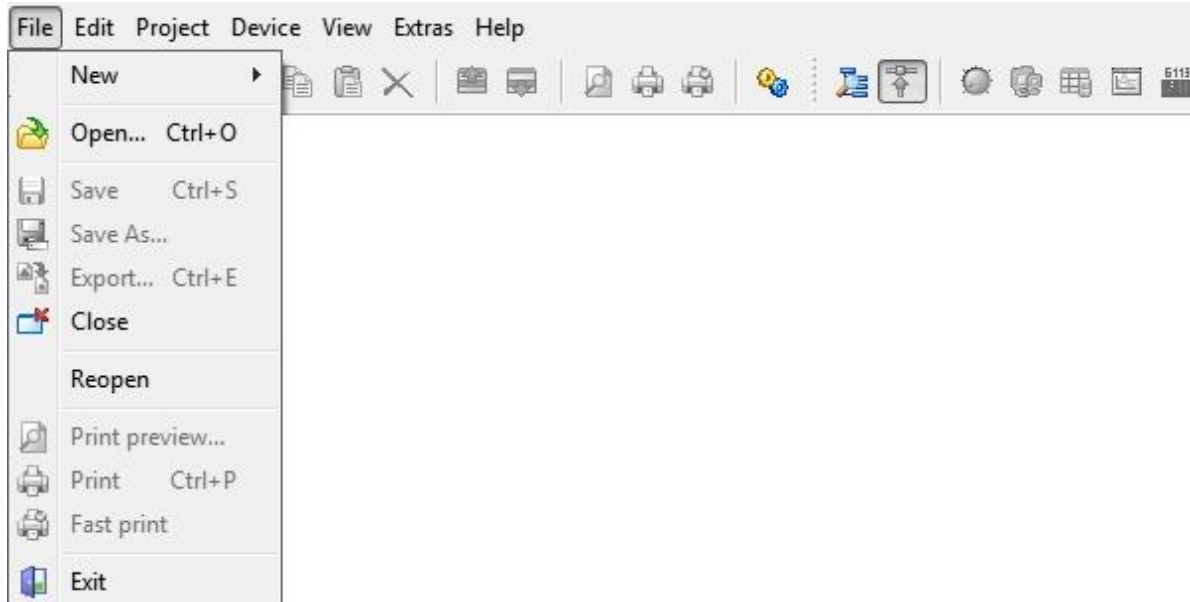
- 2.2 "Main menu"
- 2.3 "Toolbars"
- Working Area
- 2.4 "View "Project""
- 2.5 "View "Messages""
- 2.6 "View "Remote""












### 2.2 Main menu

The main menu is the central place for all actions of application. All editor windows register their window-specific actions there. The actions are divided in categories.

#### 2.2.1 Category "File"



Name of action	Shortcut	Icon	Description
New dataset			The action opens the parameter window for a new device. The user must select the desired device in a previous window.
New macro			The action opens the macro editor with a new document. If the macro window is already opened, the user can store the current document.  <b>Attention:</b> In the current version, only one macro window can be opened!
PLC program			The action opens the PLC editor with an empty document. If a PLC program is already opened, the user can store the current document.
Open	Ctrl + O		The action opens the file choice dialogue in order to open a stored document. The user selects a document type with the file filter, and selects the file afterwards. The following types are supported: <ul style="list-style-type: none"> <li>Parameter files (*.ndbx, *.db (V1.27))</li> <li>Scope files (*.scox, *.sco (V1.27))</li> <li>Macro files (*.ncmx, *.ncm (V1.27))</li> <li>PLC files (*.awlx, *.awl, *.nstx)</li> </ul>
Save	Ctrl + S		The action stores the current document. The action is passed on to the active editor window and implemented there. Depending upon the type of editor, different operations can be implemented.
Save as...			The action stores the current document with a new name. The action is passed on to the active editor window and implemented there. Depending upon the type of editor, different operations



Name of action	Shortcut	Icon	Description
			can be implemented.
Export	Ctrl + E		The action exports the data active editor windows into a file. The action is passed on to the active editor window and implemented there. Depending upon the type of editor, different operations can be implemented.
Reopen			The action contains a submenu in which the opened last documents are listed. History is limited to 5. When clicking on one of the files, it is opened again.
Print	Ctrl + P		The action is passed on to the active editor window and implemented there. Depending upon the type of editor, different operations can be implemented. This action is deactivated if no editor window is opened or the editor does not support the action.
Print preview...			The action opens a print preview for the active editor. Depending upon editor, the printing preview can be differently developed. This action is deactivated if no editor window is opened or the editor does not support the action.
Quit			The action closes application.








### Information

An action is deactivated if no editor window is opened or if the editor does not support the action.

### 2.2.2 Category "Edit"



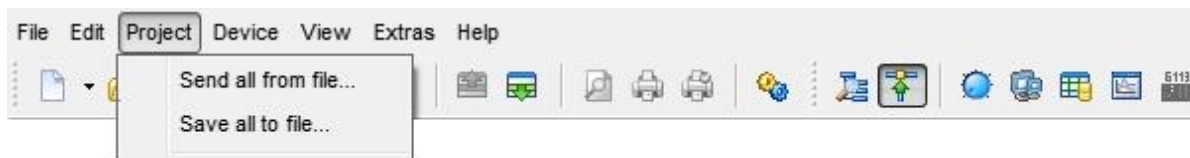
Name of action	Shortcut	Icon	Description
Undo	Ctrl + Z		The action undoes the last action. The action is passed on to the active editor window and implemented there. Depending upon the type of editor, different operations can be implemented.
Cut	Ctrl + X		The action cuts the selected object and copies it into the clipboard. The action is passed on to the active control member and implemented there. Depending upon the type of editor, different operations can be implemented.



Name of action	Shortcut	Icon	Description
Copy	Ctrl + C		The action copies the selected object into the clipboard. The action is passed on to the active control member and implemented there. Depending upon the type of editor, different operations can be implemented.
Paste	Ctrl + V		The action copies the contents of the clipboard to the selected position. The action is passed on to the active control member and implemented there. Depending upon the type of editor, different operations can be implemented.  <b>Note:</b> The action is deactivated if the current control element does not support this action or the contents of the clipboard cannot be inserted.
Delete	Ctrl + Del		The action deletes the selected object. The action is passed on to the active control and implemented there. Depending upon the type of editor, different operations can be implemented.
Select all	Ctrl + A		The action selects all objects of the active control.
Replace...	Ctrl + H		The action searches for the indicated text and replaces these then by other text. In a dialogue, the appropriate option can be adjusted.
Up	Ctrl + U		The action shifts the selected object one position upward.
Down	Ctrl + D		The action entry shifts the selected object one position downward.

### Information

An action is deactivated if no editor window is opened or if the editor does not support the action.

### 2.2.3 Category "Project"

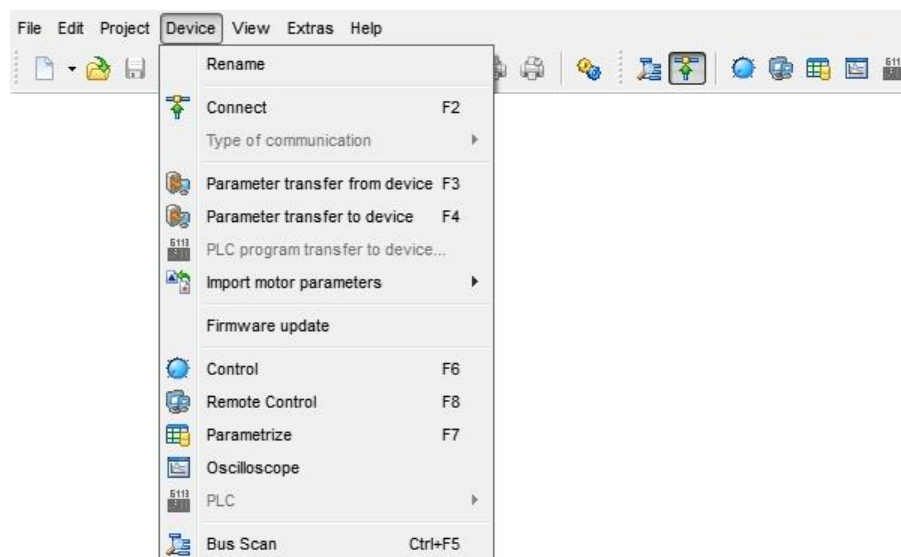








Name of action	Shortcut	Icon	Description
Alles speichern in Datei...			The action reads all parameters of devices and saves it in a file.
Alles senden aus Datei...			The action opens a file and sends the stored parameters to the devices.





### Information

An action is deactivated if no editor window is opened or if the editor does not support the action.

### 2.2.4 Category "Device"



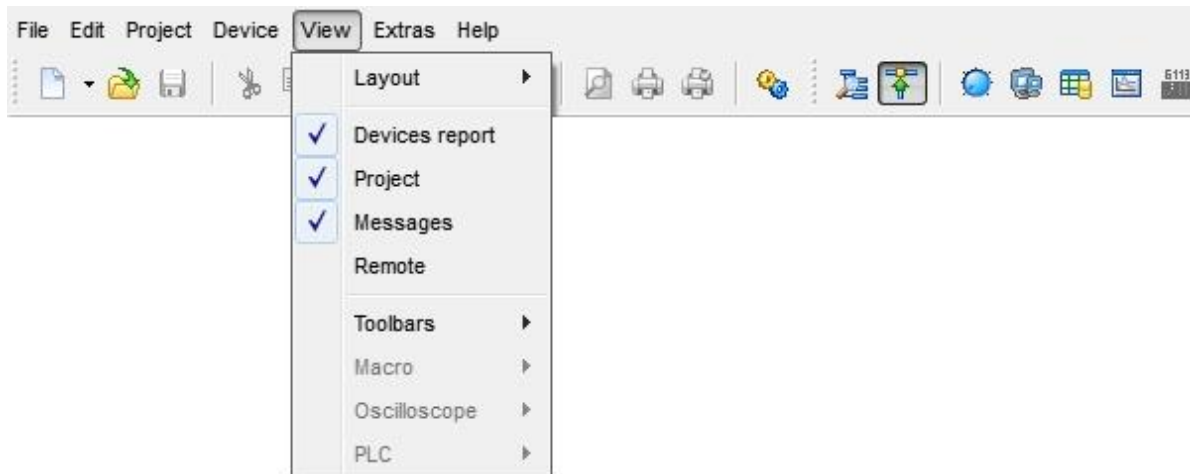
Name of action	Key combination	Picture	Description
Rename			With the aid of this menu item the user can change the name of the highlighted frequency inverter.
Connect	F2		This menu item connects or disconnects the connection to the highlighted frequency inverter.
Communication type/serial USS			This menu item sets the communication module to "serial USS". The device list is deleted if the communication type is changed!
Communication type/Ethernet			This menu item sets the communication module to "Ethernet". The frequency inverter list is deleted if the communication type is changed!
Parameter upload from frequency inverter	F3		This menu item starts the upload of parameters from the frequency inverter to the PC.
Parameter download to device	F4		This menu item starts the upload of parameters from the PC to the frequency inverter.
Transfer PLC program to device			This menu item transfers a saved PLC program to the selected frequency inverter.
Import motor parameters			This function enables motor data to be imported from an external source. If the user has selected a motor parameter file (* csv) in the file selection dialogue, all of the motors which this file contains are listed. Select a data record from the list and confirm with OK. The parameters will then be transferred to the selected frequency inverter. If the parameter window is open, the values in the parameter window are imported and not transferred to the frequency inverter. Transfer of the parameters must be carried out separately.
Update firmware			This menu item starts the firmware upload program.
Control	F6		This menu item opens the "Control" window of the highlighted frequency inverter in the work area. If the window has already been opened, it is brought into the foreground.
Remote control	F8		This menu item opens the "Remote Control" window of the highlighted in the "View and Control" window. If the window has already been opened, it is brought into the

Name of action	Key combination	Picture	Description
			foreground.
Parameter setup	F7		This menu item opens the "Parameters" window of the highlighted frequency inverter in the work area. If the window has already been opened, it is brought into the foreground.
Oscilloscope			This menu item opens the "Oscilloscope" window of the highlighted frequency inverter in the work area. If the window has already been opened, it is brought into the foreground.
PLC			This menu item opens the "PLC" window of the highlighted frequency inverter in the work area. If the window has already been opened, it is brought into the foreground.
Bus scan	Ctrl + F5		This menu item performs a network scan for the selected communication module.  <b>Notice:</b> During a network scan all inverters are removed from the list of frequency inverters and all device-specific windows are closed.

### Information

An action is deactivated if no editor window is opened or if the editor does not support the action.

### 2.2.5 Category "View"



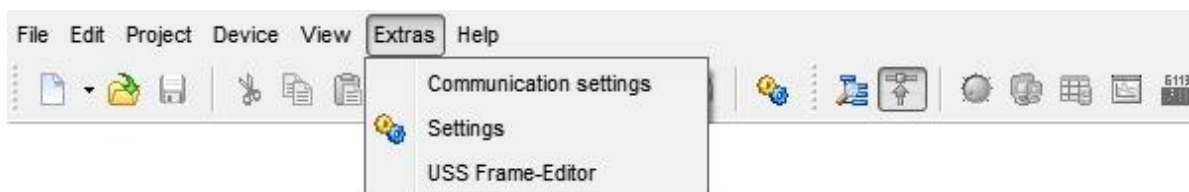


Name of action	Shortcut	Description
Layout -> Standard		The action returns the standard layout for all views. The position of the editor windows is not changed.
Layout -> Standard all windows		The action returns the standard application layout for all windows including the work area.
Device report		The action closes or opens the device report.
2.4 "View "Project""		The action closes or opens the view "Project".
2.5 "View "Messages""		The action closes or opens the view "Log".
2.6 "View "Remote""		The action closes or opens the view "Remote control".
Toolbar->Standard		The action closes or opens the toolbar "Standard".
Toolbar->Device		The action closes or opens the toolbar "Device".
Toolbar ->Start		The action closes or opens the toolbar "Start".
Macro		The action opens a submenu. In this submenu, all special actions of the macro editor are listed. The state and the execution of the actions managed by the active macro editor. If no window is active, all actions are deactivated.
Oscilloscope		The action opens a submenu. In this submenu, all special actions of the oscilloscope are listed. The state and the execution of the actions managed by the active oscilloscope. If no window is active, all actions are deactivated.

### Information

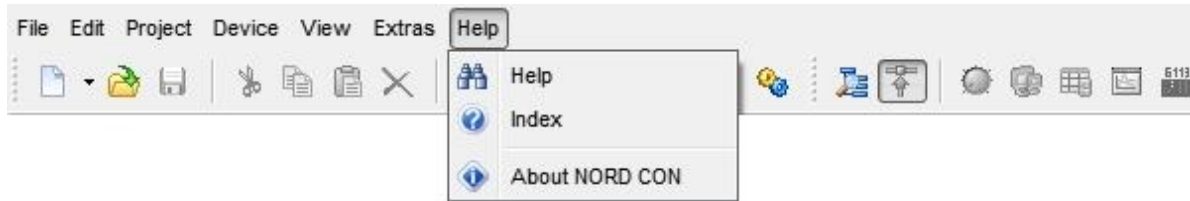
An action is deactivated if no editor window is opened or if the editor does not support the action.

### 2.2.6 Category "Extras"



Name of action	Shortcut	Description
Communication settings		The action opens a window to edit the communication settings from the selected communication module.
Settings		The action opens a window to edit the global settings of the program.
Log		The action opens a submenu. In this submenu all special actions of the view "log" are listed. The state and the execution of the actions managed by the view.

### 2.2.7 Category "Help"







Name of action	Shortcut	Description
Help	F1	The action opens online help and selects the register map "Contents".
Index		The action opens online help and selects the register map "Index".
About NORD CON		The action opens a dialogue with the program information.












## 2.3 Toolbars

In the toolbars, the most common actions are available for fast access. By clicking the appropriate symbol in the bar with the mouse, the desired action is specified.



The following toolbars are available:








### 2.3.1 Standard

Name of action	Icon	Description
New data set		The action opens the parameter window for a new device. Before the user can open the dialogue, the device must be selected.
New Macro		The action opens the macro editor with an empty document. If a macro is already open, the user can store the current document.  Attention: In the current version, only one macro window can be opened!
New PLC program		The action opens the PLC editor with an empty document. If a program is already opened, the user can store the current document.
Open		The action opens the file dialog in order to open a stored document. The user selects a document type with the file filter and select the file afterwards. The following types are supported: <ul style="list-style-type: none"> <li>Parameter dataset V1.27 (*.db)</li> <li>Parameter dataset (*.ndbx)</li> <li>Scope-File (*.sco)</li> <li>Scope-File V2.1 (*.scox)</li> <li>Macro (*.ncmx)</li> <li>Macro V1.27 (*.ncm)</li> <li>PLC Program (*.awl)</li> </ul>








Name of action	Icon	Description
Save		The action stores the current document. The action is passed on to the active editor window and implemented there. Depending upon the type of editor, different operations can be implemented.
Cut		The action cuts the selected object and copies it into the clipboard. The action is passed on to the active control element and implemented there. Depending upon the type of editor, different operations can be implemented.
Copy		The action copies the selected object into the clipboard. The action is passed on to the active control element and implemented there. Depending upon the type of editor, different operations can be implemented.
Paste		The action copies contents of the clipboard to the selected position. The action is passed on to the active control member and implemented there. Depending upon the type of editor, different operations can be implemented.  <b>Note:</b> The action is deactivated if the current control element does not support this action or the contents of the clipboard cannot be inserted.
Delete		The action deletes the selected object. The action is passed on to the active control member and implemented there. Depending upon the type of editor, different operations can be implemented.
Up		The action shifts the selected object a position upward.
Down		The action shifts the selected object a position downward.
Preview		The action opens a print preview for the active editor. Depending upon editor, the printing preview can be differently developed. This action is deactivated if no editor window is opened or the editor does not support the action.
Print		The action prints the content from the active editor. This action is deactivated if no editor window is opened or the editor does not support the action.
Fast print		The action prints the content from the active editor without the print dialog. This action is deactivated if no editor window is opened or the editor does not support the action.
Settings		The action opens a window to edits the global settings of the program.

### 2.3.2 Device

Name of action	Icon	Description
Bus scan		The action implements a network scan for the selected communication module.  <b>Note:</b> With a network scan, all devices are removed from the device list and all device-specific windows are closed!
Connect		The action connects or disconnects the connection to the selected device.

Name of action	Icon	Description
Control		The action opens the "Control" window of the selected device in the work area. If the window was already opened, it is brought into the foreground.
Remote		The action opens the "Remote" window of the selected device. If the window was already opened, it is brought into the foreground.
Parameterise		The action opens the "Parameter" window of the selected device in the work area. If the window was already opened, it is brought into the foreground.
Oscilloscope		The action opens the "Oscilloscope" of the selected device in the work area. If the window was already opened, it is brought into the foreground.
PLC		The action opens the PLC editor of the selected device in the work area. If the window was already opened, it is brought into the foreground.
Upload parameters from device		The action uploads the parameters from the device to the PC.
Download parameters to device		The action downloads the parameters from the PC to the device.

### 2.3.3 Start

Name of action	Shortcut	Icon	Description
PLC settings			The action opens the settings of the PLC.
Compile	Shift + F7		The action starts the translation of a PLC program.
Programming	Shift F8		The action loads a PLC program to the Device.
Run	F9		The action runs a PLC program or a macro. The action is passed on to the active editor window and implemented there. Depending upon the type of editor, different operations can be implemented.
Cancel	F11		The action terminates running a PLC program or macro. The action is passed on to the active editor window and implemented there. Depending upon the type of editor, different operations can be implemented.
Next	F12		The action executes the next instruction. The action is passed on to the active editor window and implemented there. Depending upon the type of editor, different operations can be implemented.
Debug	Shift + F5		The action runs the PLC program with the debug mode.

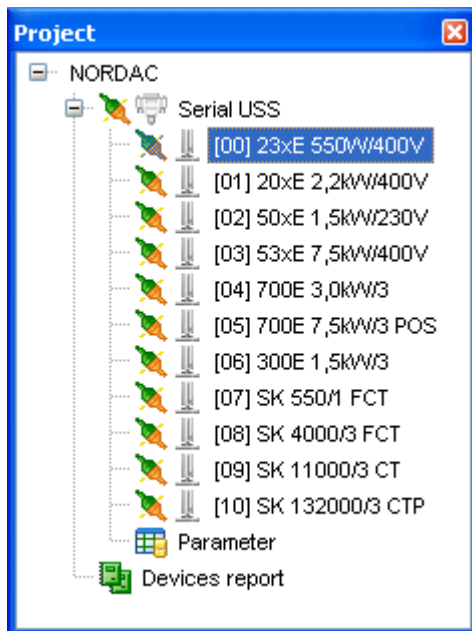
### Information

An action is deactivated if no editor window is opened or if the editor does not support the action.



## 2.4 View "Project"

In View "Project", all devices of the project are displayed in a tree structure. It can be closed or opened with the main menu option "View->Project ". With the help of the mouse, you can navigate between the individual devices. If the view is selected, you can additionally select a device with the arrow keys "up" and "down". If the pointer of mouse is over a device entry, a reference about the type of device and fieldbus address is indicated. After the selection of a device, the user can execute all actions with the tool bar as well as the popup menu. If an action is shaded grey, the selected devices do not

support. The 2.4.1 "Structure of context menu" is opened by clicking the right mouse button in the view.

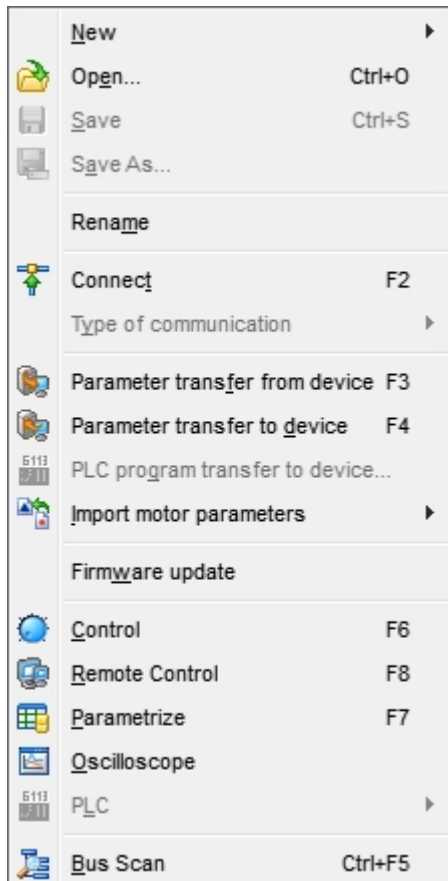


#### Status of device

-  The connection to the device is online
-  The connection to the device is offline

#### 2.4.1 Structure of context menu

The display shows the context menu of the project view. The menu always relates to the marked node in the project tree.









Name of action	Key combination	Description
New parameter set		This menu item opens the parameter window for a new device. The user must select the required device in a dialogue beforehand.
New macro		This menu item opens the macro editor with an empty document. If the macro editor was already open, the user is asked to save the current document.  <b>Attention:</b> Only 1 macro editor can be open in the current version!
New PLC program		This menu item opens the PLC editor with an empty document. If the window was already open, the user is given the opportunity to save the current document.
Open	CTRL + O	This menu item opens the PLC editor with an empty document. If the window was already open, the user is given the opportunity to save the current document.
Save	CTRL + S	This menu item saves the current document. The action is forwarded to the active editor window and carried out there. Depending on the editor type, different operations can be carried out there.
Save as ....		This menu item saves the current document under a new name. The action is forwarded to the active editor window and carried out there. Depending on the editor type, different operations can be carried out there.
Rename		This action opens an input field for changing the device name.
Connect	F2	This action connects or disconnects the connection to the highlighted device.



Name of action	Key combination	Description
Communication type/serial USS		This menu item sets the communication module to "serial USS". The device list is deleted if the communication type is changed!
Communication type/Ethernet		This menu item sets the communication module to "Ethernet". The device list is deleted if the communication type is changed!
Parameter upload from frequency inverter	F3	This action starts the upload of parameters from the device to the PC.
Parameter download to device	F4	This menu item starts the download of parameters from the PC to the device.
Update firmware		This action starts the program for uploading the firmware.
Control	F6	This action opens the "Control" window of the highlighted device in the work area. If the window has already been opened, it is brought into the foreground.
Remote control	F8	This action opens the "Remote Control" window of the highlighted device in the "View and Control" window. If the window has already been opened, it is brought into the foreground.
Parameter setup	F7	This action opens the "Parameter" window of the highlighted device in the work area. If the window has already been opened, it is brought into the foreground.
Oscilloscope		This action opens the "Oscilloscope" window of the highlighted device in the work area. If the window has already been opened, it is brought into the foreground.
PLC		This action opens the "PLC" window of the highlighted device in the work area. If the window has already been opened, it is brought into the foreground.
Bus scan	CTRL + F5	This action triggers a new bus scan.  <b>Attention:</b> During a bus scan, all devices are removed from the device list and all device-specific windows are closed!

## 2.5 View "Messages"

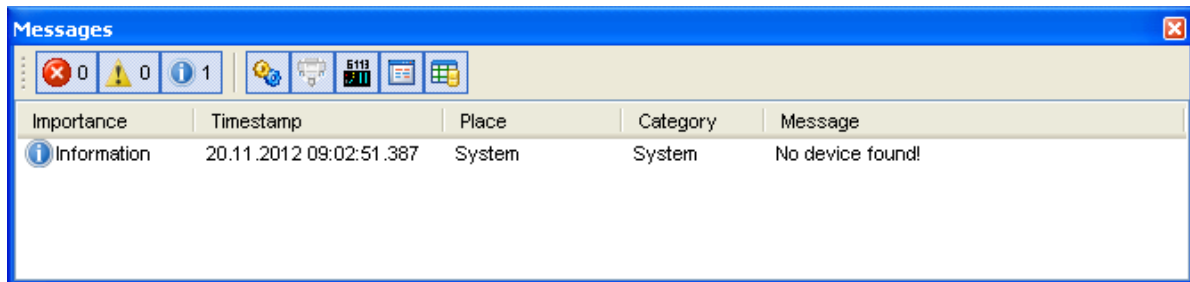
The view contains a list of all "NORD CON" messages. The entries are displayed by default as time ascending. The order can be adjusted by clicking on a column header. Following filters are available for filtering:

Filter	Icon	Description
Error		When this filter is enabled, displays all errors. In addition, it shows the number of errors in the caption of the button.
Warning		When this filter is enabled, all warnings are displayed. The number of warnings is displayed in the caption of the button.
Information		When this filter is enabled, all information will be displayed. The number of information items is displayed in the caption of the button.
System		When this filter is enabled, all messages of the "System" category are displayed.
Communication		When this filter is enabled, all messages of the "Communications" category are displayed.
PLC		When this filter is enabled, all messages of the "PLC" category are displayed.



Macro		When this filter is enabled, displays all messages in the "Macro" category.
Parameter		When this filter is enabled, displays all messages in the "Parameter" category.

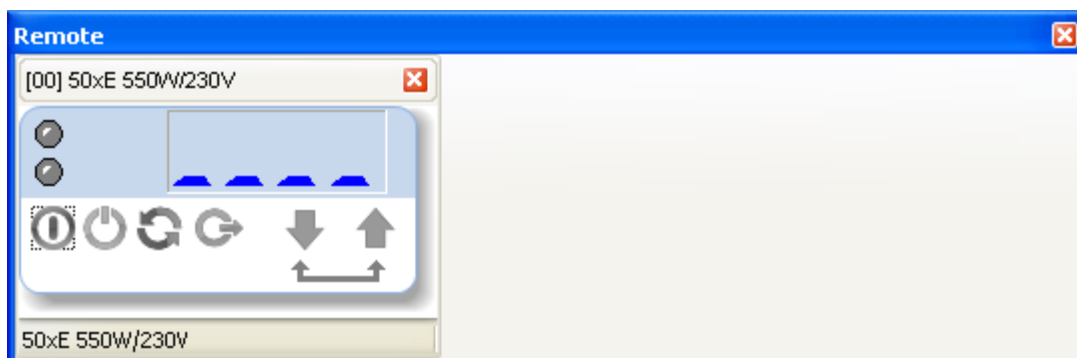
The messages can be saved or deleted via the popup menu (right mouse button). These actions can be carried out via the main menu ("Extras/Messages").



Name of action	Description
Delete	The action deletes the list.
Save	The action stores the entries into a file.

### 2.6 View "Remote"

The view "remote" contains all windows of the function „2.6 "View "Remote"“. The view opens automatically when opening the first window and closes after closing the latest. The view can be docked or undocked like all views to the work area. If the view was closed by the user, it can be opened by the action "Remote" again. The new windows are always docked to the left side of the last window. With the help of the mouse, it can be undocked or docked again. If the view is displayed for the first time (View->Remote), all remote windows are automatically opened.



#### Information

#### Remote

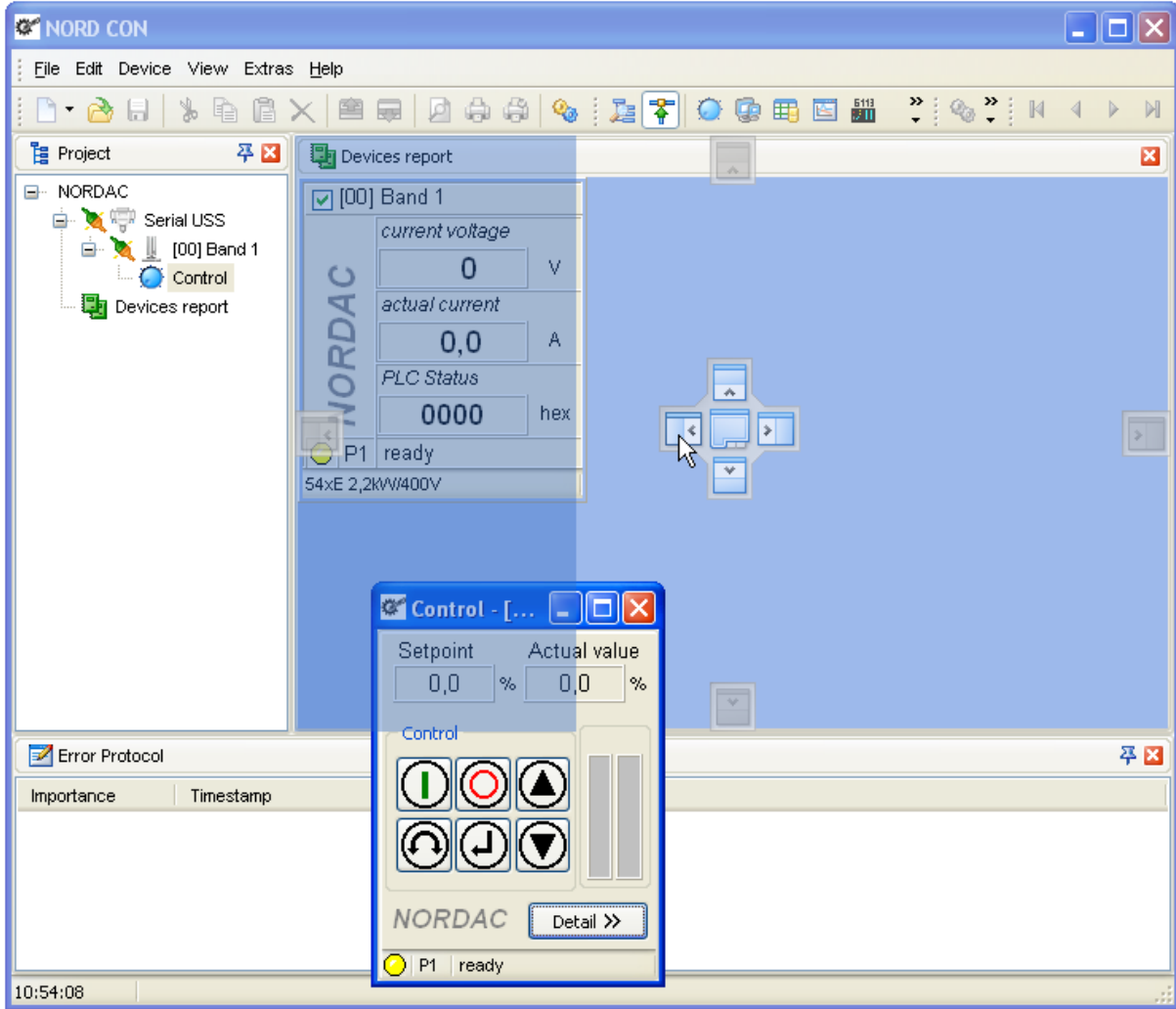
Windows of type "Remote" can be docked only into the view "Remote".

## 2.7 Docking and Undocking

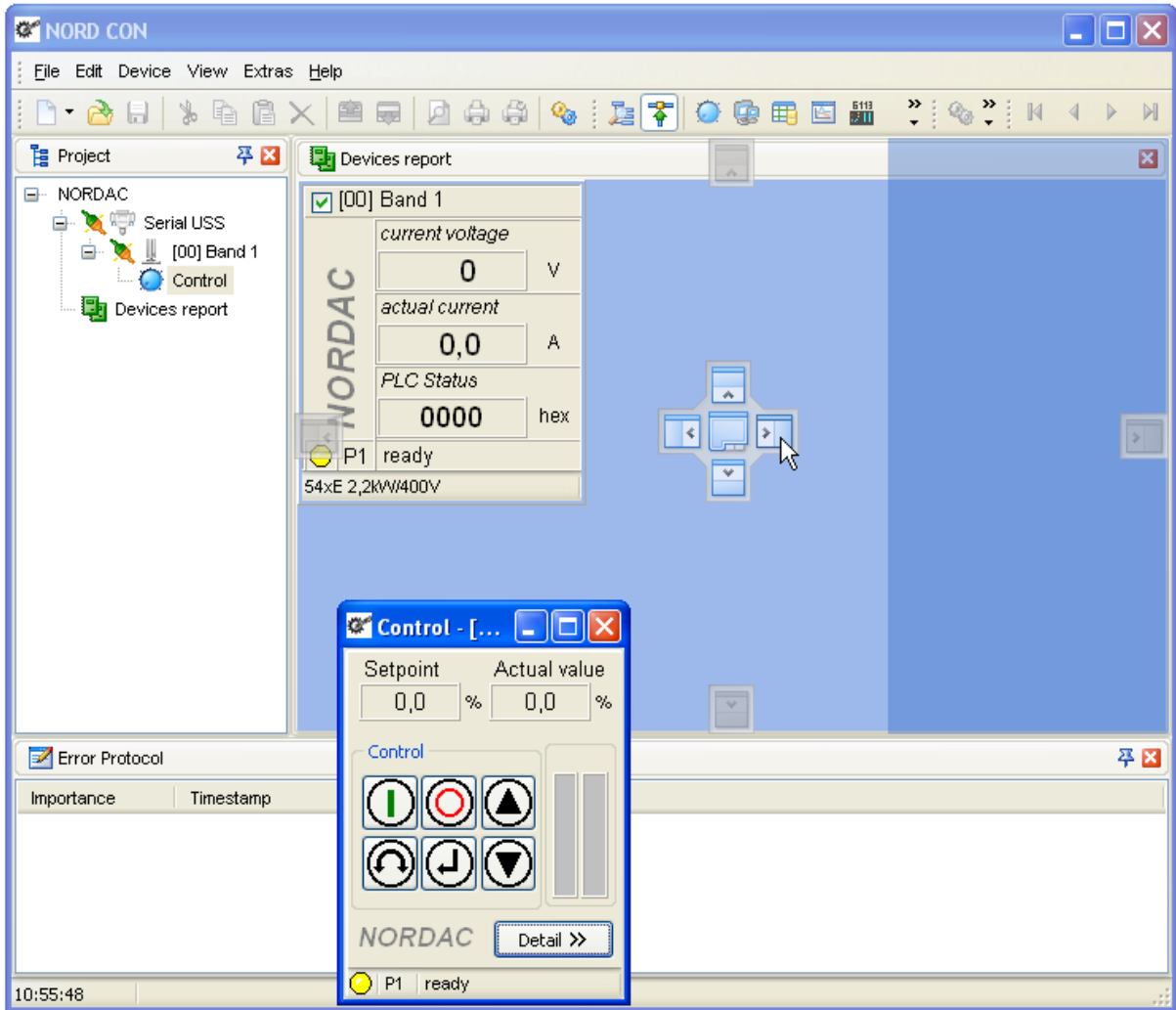
With the new design of NORD CON, the user has the possibility to adapt the layout of the window to their own requirements. In principle, you can undock each view and editor window and position them freely on the screen. For this, the user must press the left mouse button over the title border and move the coloured rectangle to the desired position. After releasing the mouse button, the view or editor windows remains in those positions as independent windows. With the editor windows, there is additionally the possibility - with the popup menu, which opens when clicking with the right mouse button on the title border to undock the windows. The docking functions are similar to the undocking functions. The coloured rectangle indicates in each case the current docking position.

Type of window	Rule
View of main window(e.g. Project, Logs, Remote)	The views of the main window can be docked only to the left, right and/or lower edge of the work area. Within these windows, there are no rules and the user can select the position freely.
Editor window (e.g. Macro editor, Parameter window, Oscilloscope)	The editor windows can only be docked into the work area. The adjustment is fixed however on down and/or above, or as register map.
Views of the Macro editor	The views of the macro editor can be docked only to the macro window. The adjustment here is fixed on left, right or down. Within the views, no rules are defined.
Views of the oscilloscope	The views of the oscilloscope window can be docked only to the oscilloscope window. The adjustment here is fixed on left, right or down. Within the views no rules are defined.
"Remote" windows	"Remote" windows can only be docked to the view "Remote". Here the adjustment is fixed on left.

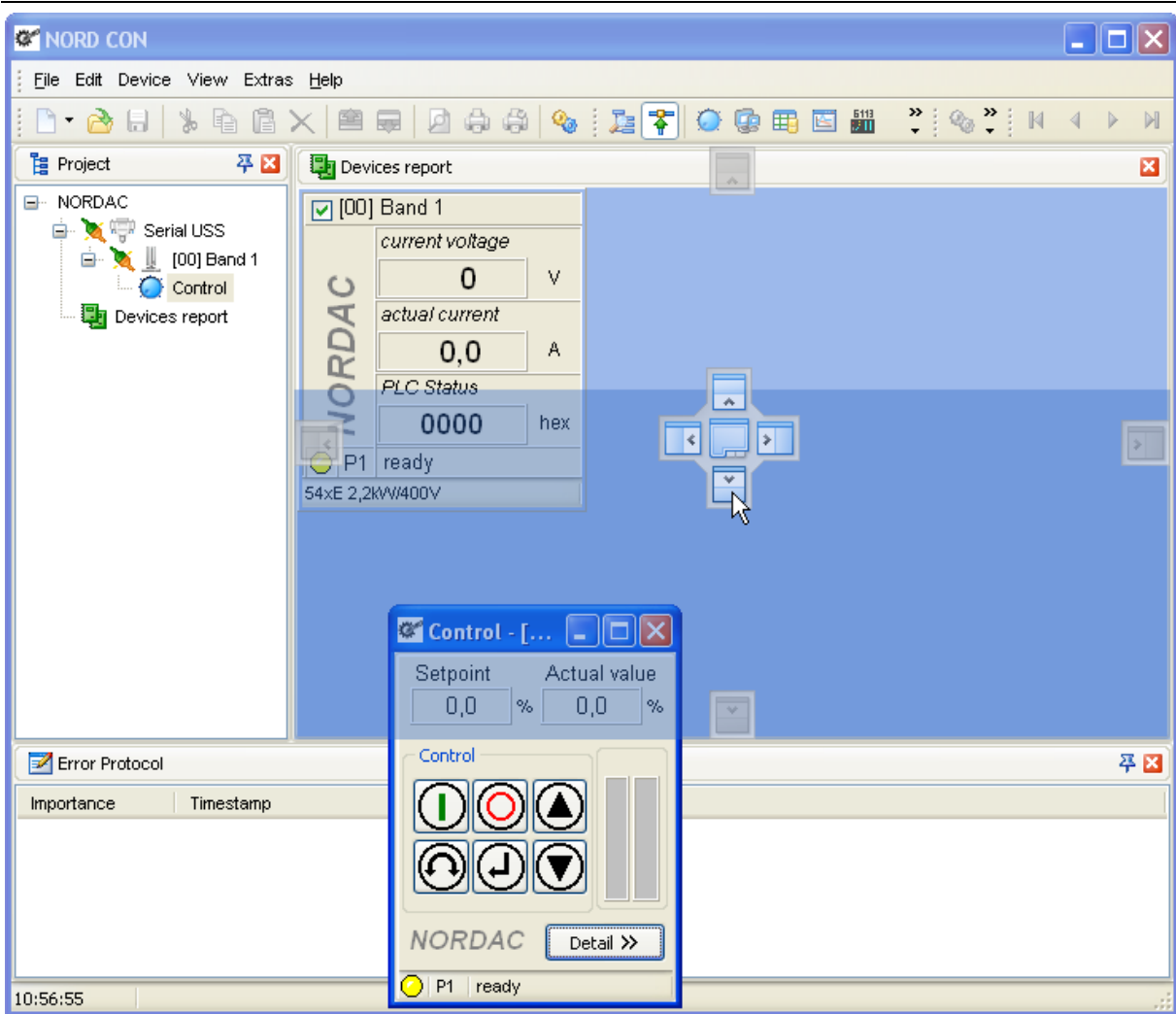
### Docking position left



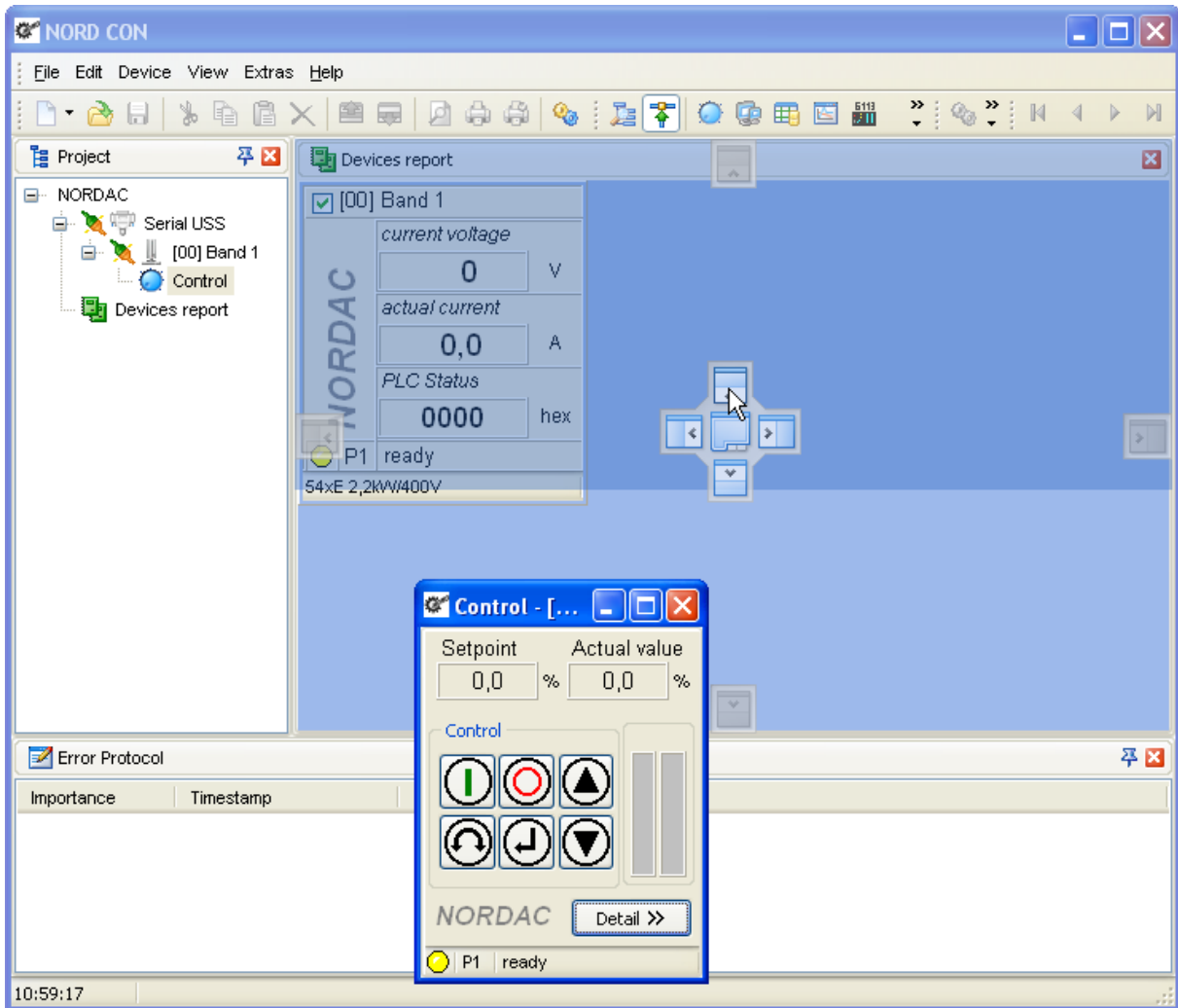
**Docking position right**



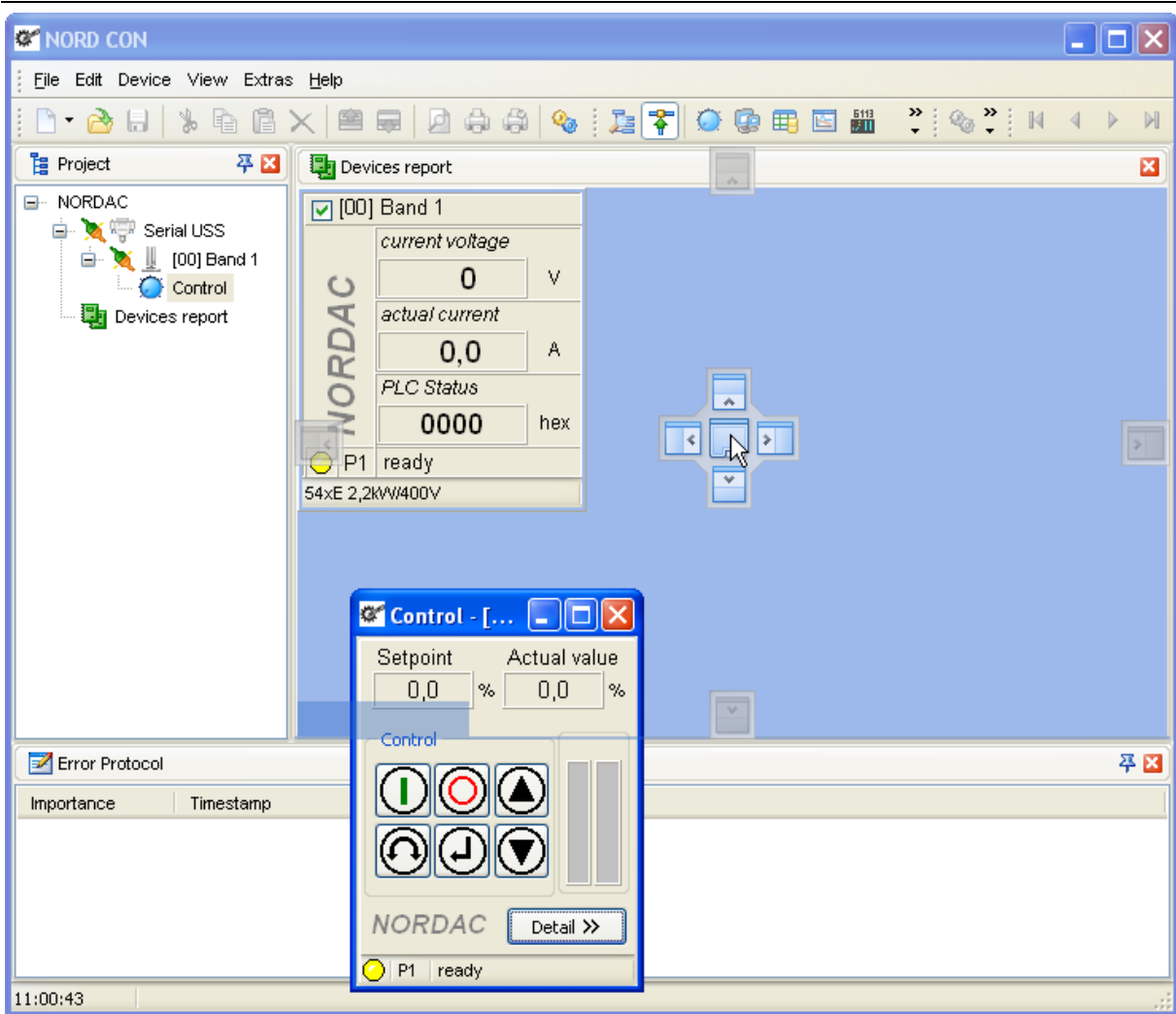
**Docking position down**



**Docking position up**



### Docking position tab





## 3 Communication

In order to start a connection to a device, you must insert the appropriate communication module in the project. After the installation, a USS communication module is configured. With the action "Parameterise" the user can modify the parameters of the module.

**Presently the following communication modules are supported:**

### 3.1 USS

#### 3.1.1 General settings

##### **Name**

The user can assign a name for the communication module in the input field.

##### **Port**

In the selection box, the user specifies the COM port of the PC to which the frequency inverter is connected.

##### **Telegram error**

In the input field, the user specifies the number of permissible telegram errors. Telegram errors occur if the content of the telegram is not correct, i.e. if the response does not correspond to the parameter entry. Usually a response is given after 2 telegrams for each order. The number of permissible telegram errors specifies how many attempts are permitted before an error message is displayed.

##### **Bus error**

In the input field, the user specifies the number of permissible bus errors. A bus error occurs if the receipt or transmission telegram has an error. The incorrect telegrams are deleted. Here, the permissible number of incorrect telegrams for which an error is generated can be set. The error tolerance should be set to an appropriately higher value for environments in which there is interference.

##### **Stop all detected devices**

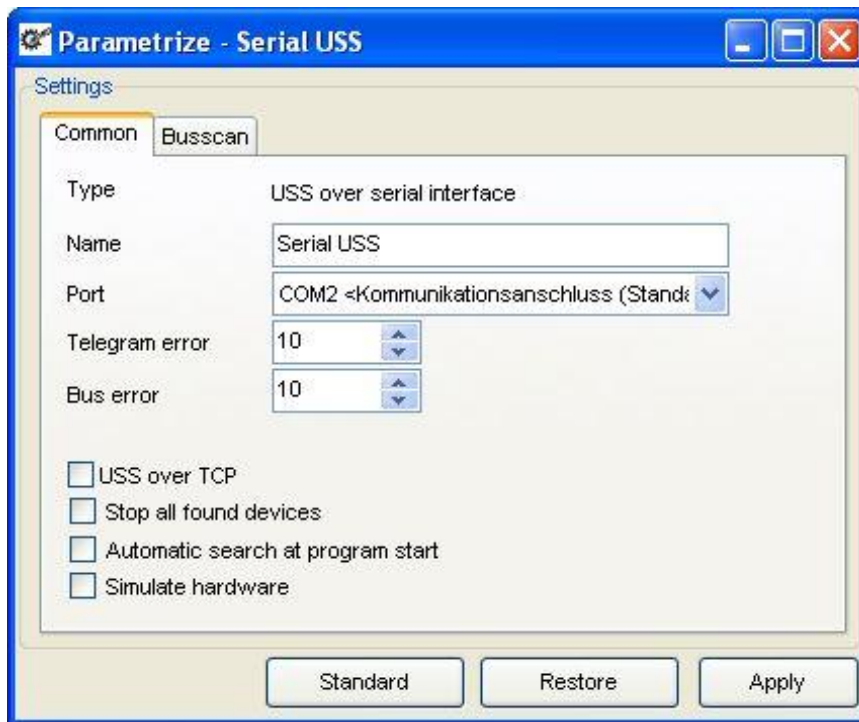
If this option is enabled, after the device search the NORD CON sends a "Disable" command to each of the devices which are detected. The device is stopped if it can be controlled via the bus (P509).

##### **Automatic device search after program start**

This option enables or disables the automatic search when the program is started. If this option is enabled, a device search is started automatically when the NORD CON is started.

##### **Simulate hardware**

With this option, the user can enable or disable simulation of a connected piece of hardware.



#### **i** Information

#### Changes

All changes only become effective after pressing the "Adopt" button. The currently valid settings can be restored with the "Restore" button.

### 3.1.2 Bus scan

#### Baud rate

In the selection box, the user can choose the communication speed of the serial interface. The same value must be chosen on the frequency inverter. When using multiple frequency inverters, the setting must be identical on all connected devices. The baud rates over 115200 Bit/s are user specific baud rates and not supported by all devices.

#### **i** Information

#### Connection problems

Older serial PC interfaces are sometimes not able to justify the accurate user specific baud rate. From this reason no connection can be made to the device.

#### Bus-Scan with all baud rates

With the action, the user activates or deactivates the bus scan with all baud rates. If the baud rate of the connected device is unknown, the search with all baud rates can find the right one to start communication.

#### Starting baud rate

In the selection box, the user can define the baud rate for start of the baud rate search.

#### Starting address

In this field, the USS address can be defined, from where the search run of NORD CON starts to find connected devices. All frequency inverters with lower address cannot be found by NORD CON.

### End address

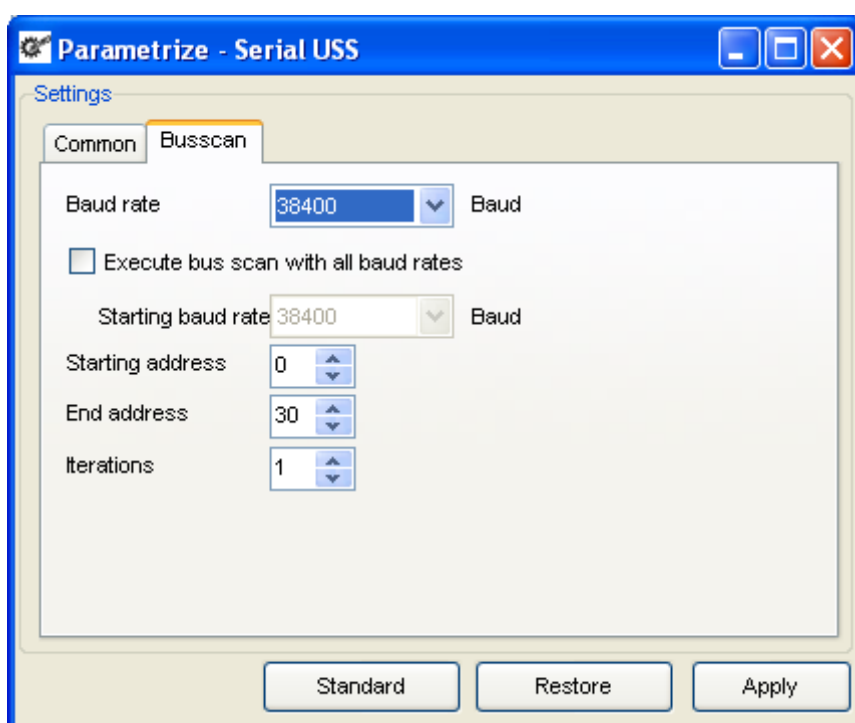
In this field, the user can define the USS address for the ending of the search for connected devices. All inverters with a higher address number cannot be found by NORD CON.

### Stop all connected devices

With the action, the user can activate or deactivate the stopping (disable) of connected devices. When this function is active, all enabled devices are stopped if the interface of the device is programmed to "bus".

### Automatic device search after start of program

With this action the user can activate or deactivate the automatic device search after the start of the program. When this function is active, NORD CON automatically starts the bus scan after the program is started.



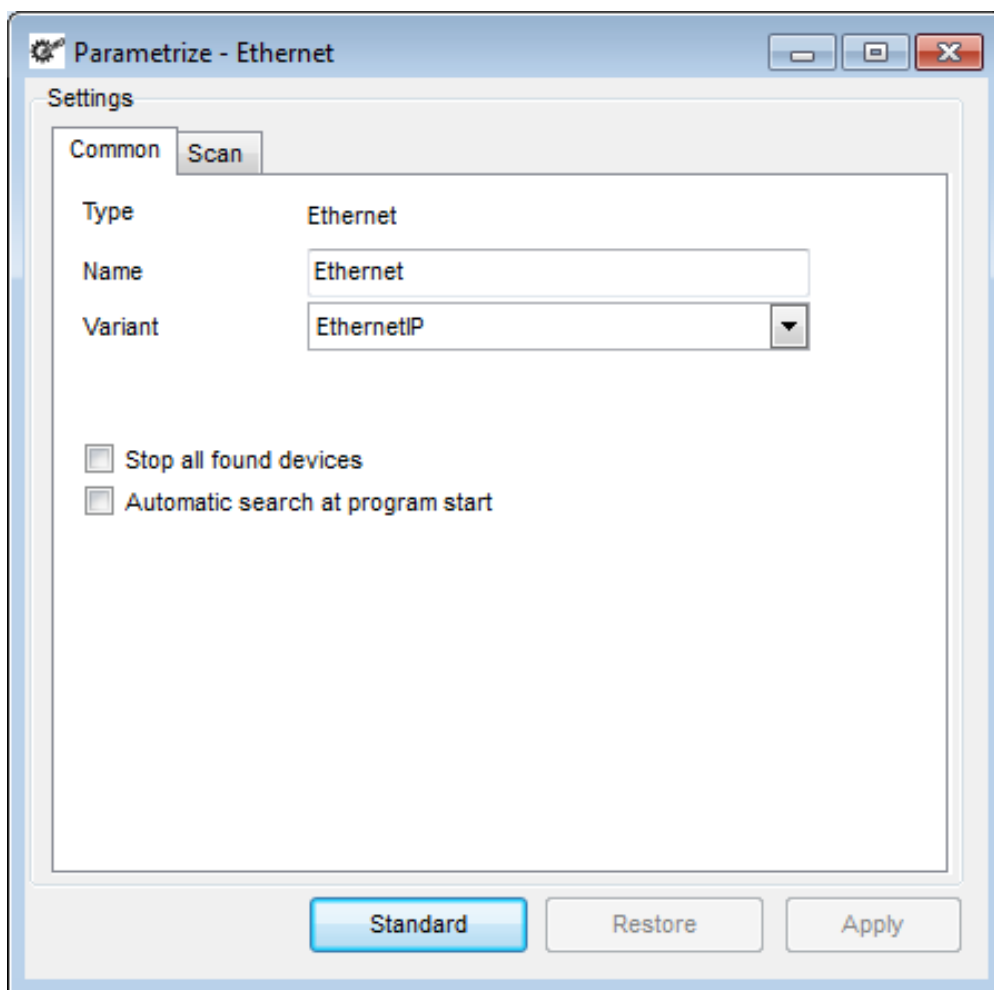
### Information

### Changes

All changes only become effective after pressing the "Adopt" button. The currently valid settings can be restored with the "Restore" button.

## 3.2 Ethernet

#### 3.2.1 General settings



##### Name

In the edit field, the user can assign a name for the communication module.

##### Kind

In the combo box the user defines the kind of communication (PROFINET, EtherCAT or EthernetIP).

##### Stop all found devices

If this option is enabled, each detected device will be sent the "Disable" command after search, provided that the device can be controlled via bus.

##### Automatic search at program start

If this option is enabled, the device search is initiated after the program start.

---

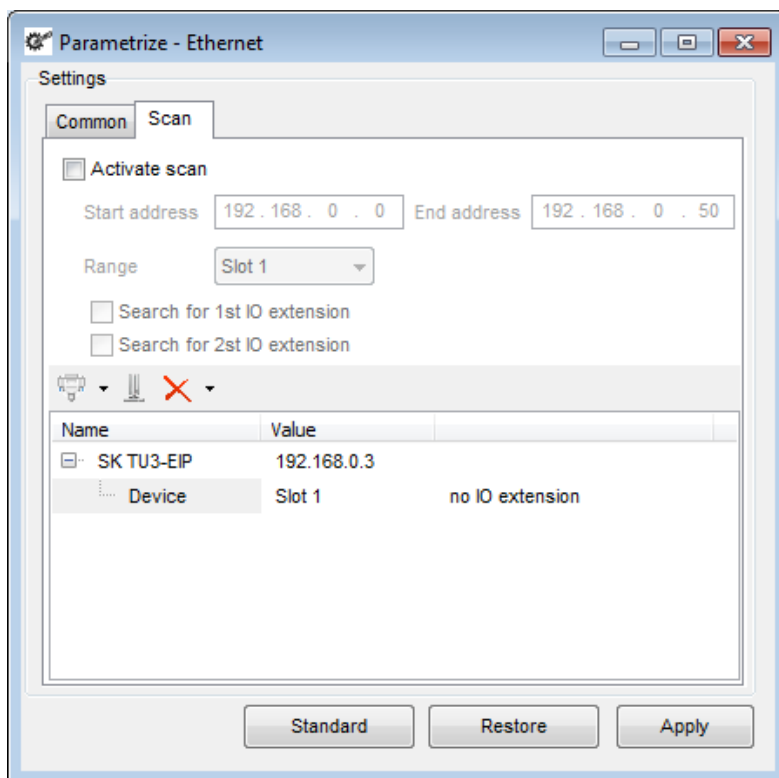
#### **i** Information

#### Changes

All changes only become effective after pressing the "Adopt" button. The currently valid settings can be restored with the "Restore" button.

---

### 3.2.2 Scan



#### Activate scan

The option defines whether the device search is enabled. If the search was activated, all IP addresses are searched for devices from the start address to the end address. If the search is disabled in a bus scan, the following configuration is used.

#### Starting address

In this edit box you have to insert the start address for the device search.

#### End address

In this edit box you have to insert the end address for the device search.

#### Add bus module

The button adds a new bus module in the device list.

#### Add device

The button adds a new device in the device list.

#### Delete

The button removes the highlighted entry in the list of devices.

#### Value - bus module (Address):

In the column you have to enter the IP address of the bus module.

**Value - device:**

In the column you have to enter the slot on the device (see following table).

Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
System bus address 32 or SK 5xxE over TU3	System bus address 34	System bus address 36	System bus address 38	System bus address 40	System bus address 42	System bus address 44	System bus address 46

**Sample:**

**Additional - device:**

In this column, the user specifies the number of IO extensions.

Bus module	Slot 1	Slot 2	Slot 3	Slot 4
SK TU3-EIP V1.2 SK TU3-PNT V1.2	SK 5xxE	not available	not available	not available
SK CU4-EIP V1.2 SK TU4-EIP V1.2 SK CU4-PNT V1.2	SK 5xxE SK 2xxE SK 19xE SK 18xE	SK 5xxE SK 2xxE SK 19xE SK 18xE	SK 5xxE SK 2xxE SK 19xE SK 18xE	SK 5xxE SK 2xxE SK 19xE SK 18xE

** Information**

**Access rights**

Please note that you need access rights for parameter changes or control via the bus module. To set it accordingly please check the data sheet of your field bus module.

** Information**

**Changes**

All changes only become effective after pressing the "Adopt" button. The currently valid settings can be restored with the "Restore" button.

## 4 Parameterization

All parameters of the frequency inverter that can be changed can also be changed by NORD CON. All of the parameters can be stored and retransmitted to the frequency inverter. Parameters which have been read out can be printed out for documentation purposes.

### 4.1 How to manipulate parameters

The selected parameter is read out and the value transferred to the 'Current Setting' box. Management of the parameters of a frequency inverter is ensured by databases. These databases can be stored, printed out or manipulated again at a later date.

#### Information

#### Menu "Parameterise"

The menu "Parameterise" is indicated only if a parameter window is marked.

NORD CON features two ways of parameter manipulation:

Action	Place	Description
New	File -> New -> Dataset	The current database is re-initialized, in other words the current and the new settings are deleted.
Open	File -> Open	Any database that was saved can be reopened.
Save	File -> Save	The current database is saved with the current name.
Save as...	File -> Save as...	The current database is saved with a new name.
Print preview...	File -> Print preview...	The current parameter settings are printed out.
Read all parameters or Read all	Parameterise -> Read -> All Parameter	All of the parameters of the frequency inverter are read out and entered into the database.
Read actual menu group	Parameterise -> Read -> Actual menu group	The parameters of the selected menu group are read out and entered into the database.
Send new settings	Parameterise -> Send -> new Values	All parameters for which a new value was entered in the 'New settings' box are transmitted to the frequency inverter. A selection is possible as to whether this operation is to be performed on all parameters or only on those belonging to the current menu group.
Send Factory settings	Parameterise -> Send -> Reset values	The settings transmitted will be the default settings of all parameters or of the parameters of the current menu group respectively
Selection Enable	Parameterise -> Selection -> Release	All of the parameters (or those included in the current menu group respectively), are enabled.
Selection Disable	Parameterise -> Selection -> no Release	None of the parameters (or of those belonging to the current menu group), are enabled.
Standard	Button "Standard"	The default value is allocated to the currently selected parameter.

Action	Place	Description
Send	Button "Send"	The value "new setting" of the current parameter is transferred to the inverter
Read	Button "Read"	The selected parameter is read out and the value transferred to the 'Current Setting' box.

With the Auto-read option the selected parameter is read out automatically.

### 4.2 Selective parameterisation

NORD CON allows for masking some parameters or other, a feature which may facilitate manipulation or serve the purpose of restricting parameter readout or transmission to those which remain unmasked, or in other words have been filtered out.

#### Information

When a filter has been activated, all operations are executed only on those parameters which are displayed.

Before any parameter can be masked the enable command must be inactivated. This can be done using the checkbox preceding the parameter, or via the 4.1 "How to manipulate parameters" menu.

The Filter box provides for the setting options mentioned below:

- **Selection only** Only the enabled parameters are displayed (i.e. where the check box preceding the parameter was clicked once).
- **No standard** Only the parameters with a value that is different from the standard setting are displayed.
- **Info parameters**
  - **Yes** Information parameters are displayed.
  - **No** Information parameters are not displayed.
- **Only** Information parameters are displayed exclusively.

### 4.3 Off-line Parameterisation

Off-line parameterisation implies that a database is manipulated which is not allocated to any frequency inverter connected.

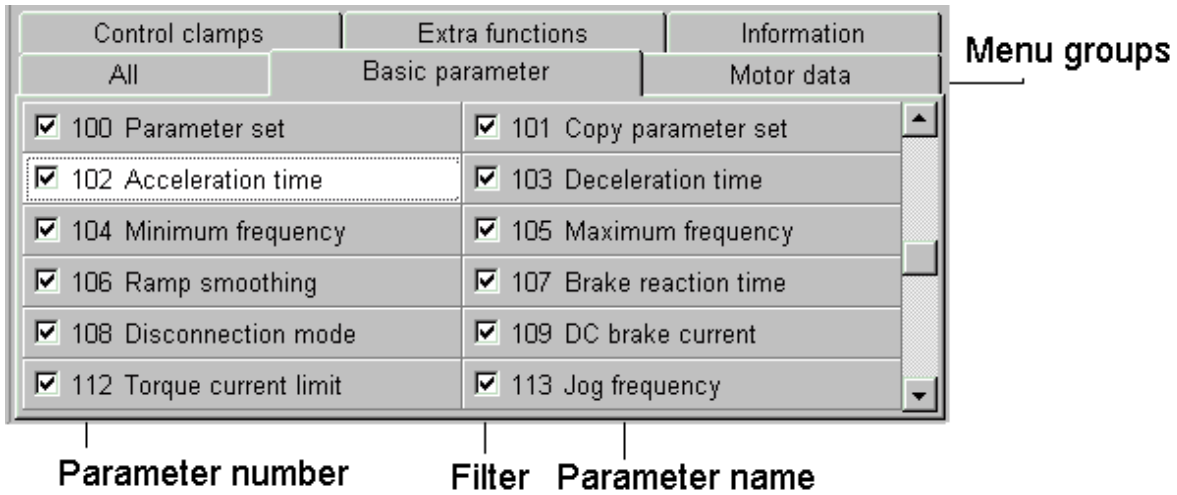
Off-line parameterisation is started via the database menu in the main window.

Name	Description
New	A new database can be created. The new database is allocated to a frequency inverter type which is set using a selection box.
Open	Any database that was read into memory can be opened and manipulated.

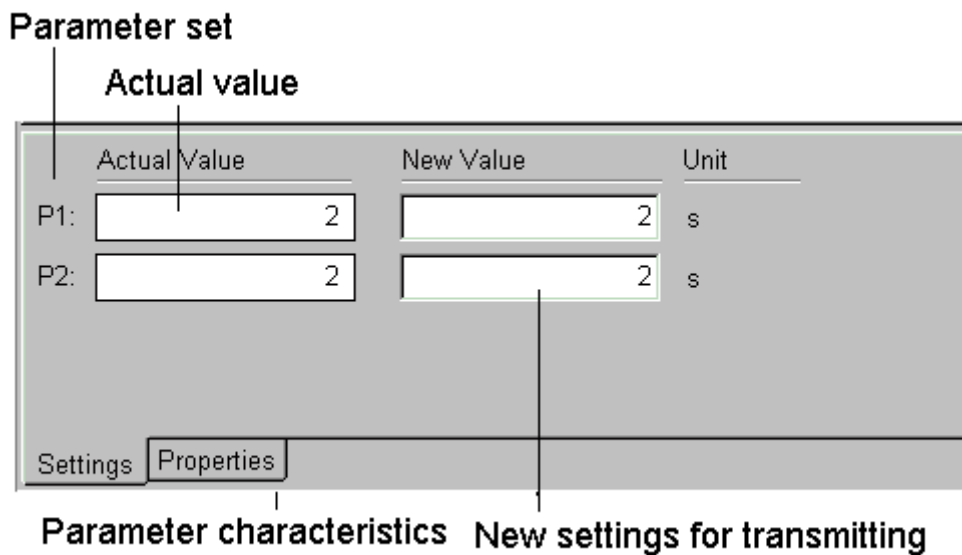


### 4.4 Parameter Viewing

Each parameter has a parameter name and a unique parameter number by which it can directly be accessed. The parameters are divided into menu groups.



Each parameter has a parameter value and parameter characteristics:



When a parameter has been selected, values of all parameter sets, if it can be set differently in the sets, are displayed.

### 4.5 Comparison report

The report shows the differences and similarities between two data records in a window. Basically, only data records in one device family can be compared. The parameters are shown in the form of a list. If two parameters differ, the line is marked with a grey bar. It is also checked whether a value differs from the standard value. If this is the case, the value is displayed in red.

---

### Information

### Save dataset

After the report has been generated, the data record can no longer be saved! For this reason, it is advisable to save the data record beforehand.

---

#### Online / Offline comparison

A device with NORD CON must be connected in order to perform the comparison. The parameter window for the device must then be opened, and it is advisable to read out all parameters. The parameter selection can be restricted even further using the filters. A report can then be generated using the “Parameter Setup -> Comparison” menu item. After calling up the function, the user must select a stored data record for the comparison. If the parameters which are read out are going to be used as a back-up, the user must subsequently save the current data record. The report is then generated and displayed.

---

### Information

The configuration of the open parameter set is used as the reference for the parameters and the standard values. If a data record that does not correspond with the configuration of the device is selected, any parameters that are not present are shown empty and marked as different.

---

#### Offline / Offline comparison

A saved or new data record must be opened in order to perform the comparison. The parameter selection can be restricted even further using the filters. Then a report can be generated using the “Parameter Setup -> Comparison” menu item. After calling up the function, the user must select a stored data record for the comparison. The report is then generated and displayed.

---

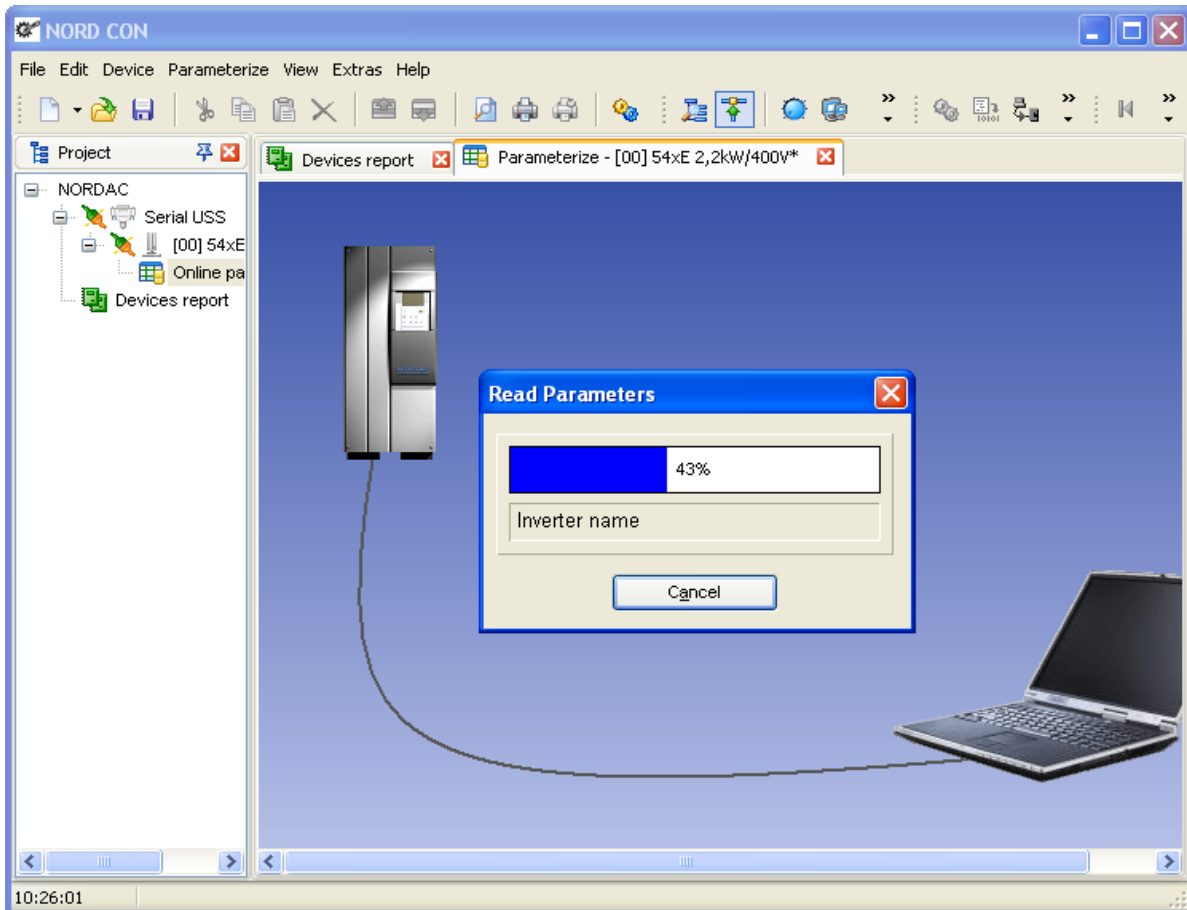
### Information

The configuration of the open parameter set is used as the reference for the parameters and the standard values. If a data record that does not correspond with the configuration of the device is selected, any parameters that are not present are shown empty and marked as different.

---

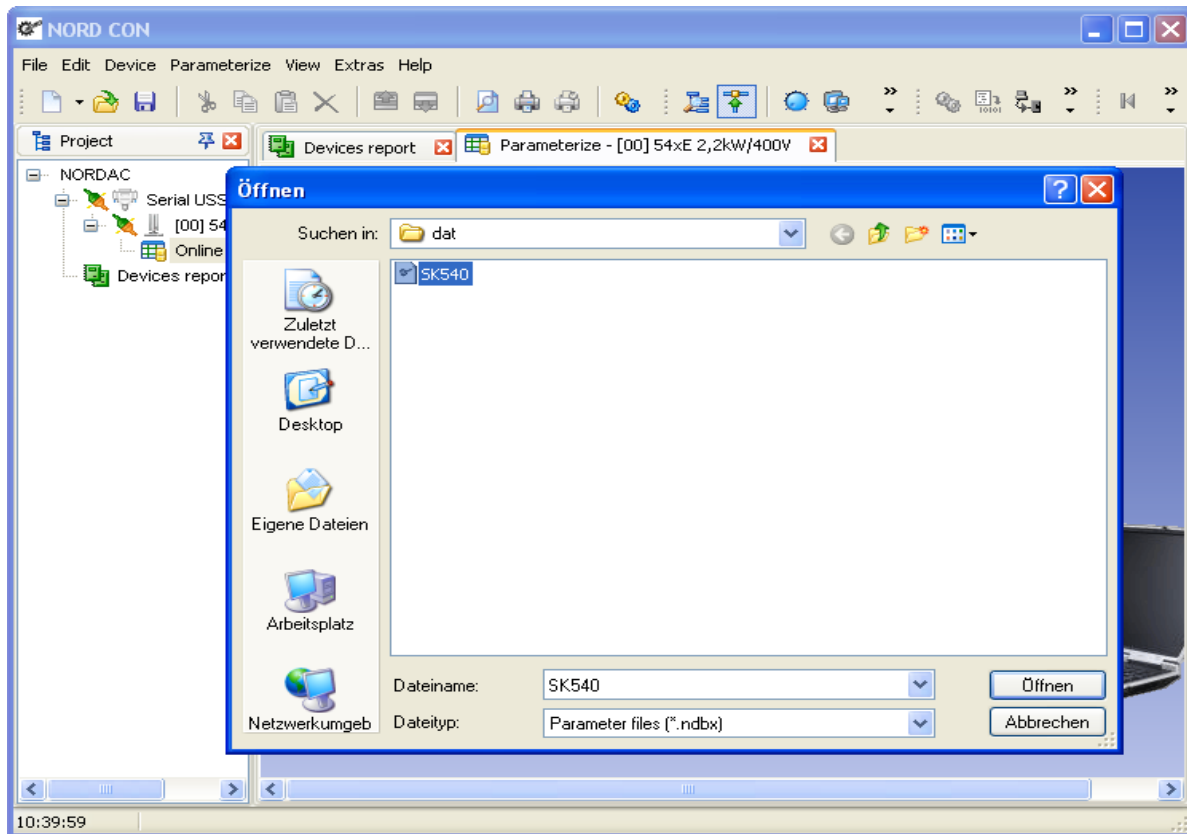
## 4.6 Parameter upload from device

The “Parameter upload from device” function loads the parameters of a device onto the PC and then saves the values in a parameter file. This action can be called up using the “Device” toolbar or via the “Device/Parameter upload from device” menu item. After executing the function, the following window opens and the upload of the parameters starts automatically. If communication errors occur during the transfer, these are displayed in the message window. At the end of the transfer the user is requested to enter a file name for the file. If the user confirms with “Save”, the parameters are saved.



#### 4.7 Parameter download to device

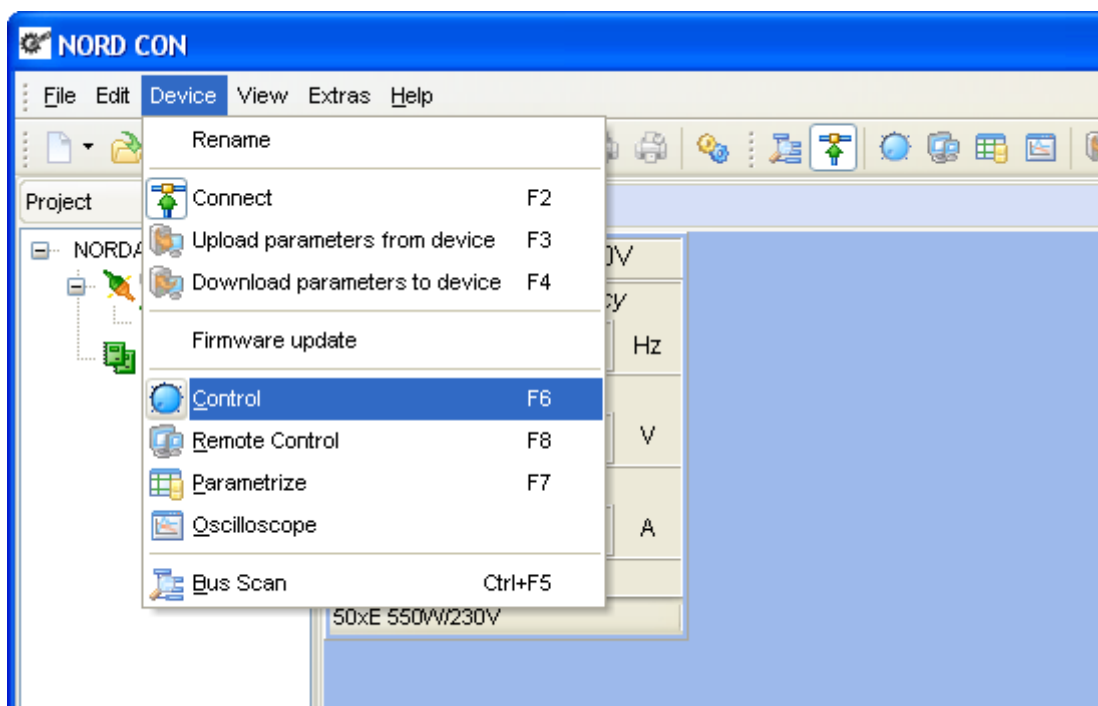
The “Parameter download to device” function opens a parameter file on the PC and transmits all values to the device. The action can be called up using the “Device” toolbar or via the “Device/Parameter download to device” menu item. After executing the function, the following window and a file selection dialogue open. The user selects a parameter file in this dialogue and confirms with “Open”. Then it is checked whether the parameter file is suitable for the selected device. If this is the case, the download is started.



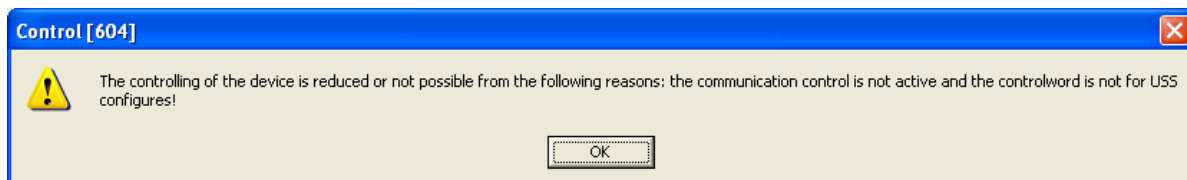
## 5 Control

### 5.1 Overview

The program NORD CON can be used to control NORD Frequency inverter. To use this function the inverter must be parametrised in the right way. Because of different settings of different inverter types the user must check the manual to find the right settings. Before the inverter can be controlled the Bus-scan must be done. After the scanning process has finished all connected inverter are displayed in the main window. Now the inverter to be controlled can be chosen by mouse click. The window „Control“ can be opened via "Device/Control (F6)" in the main menu or via popup menu (right mouse click).



Now the control configuration of the inverter is read and checked with the standard setting (setting/control/control configuration check). If the "Control" of inverter is limited or impossible there will be a warning note on the screen.



In the window "Control" there are two versions available:

- 5.2 "Standard control"      The frequency inverter can be released and the setting value can be increased or decreased. Direction change and error acknowledge is possible, too.

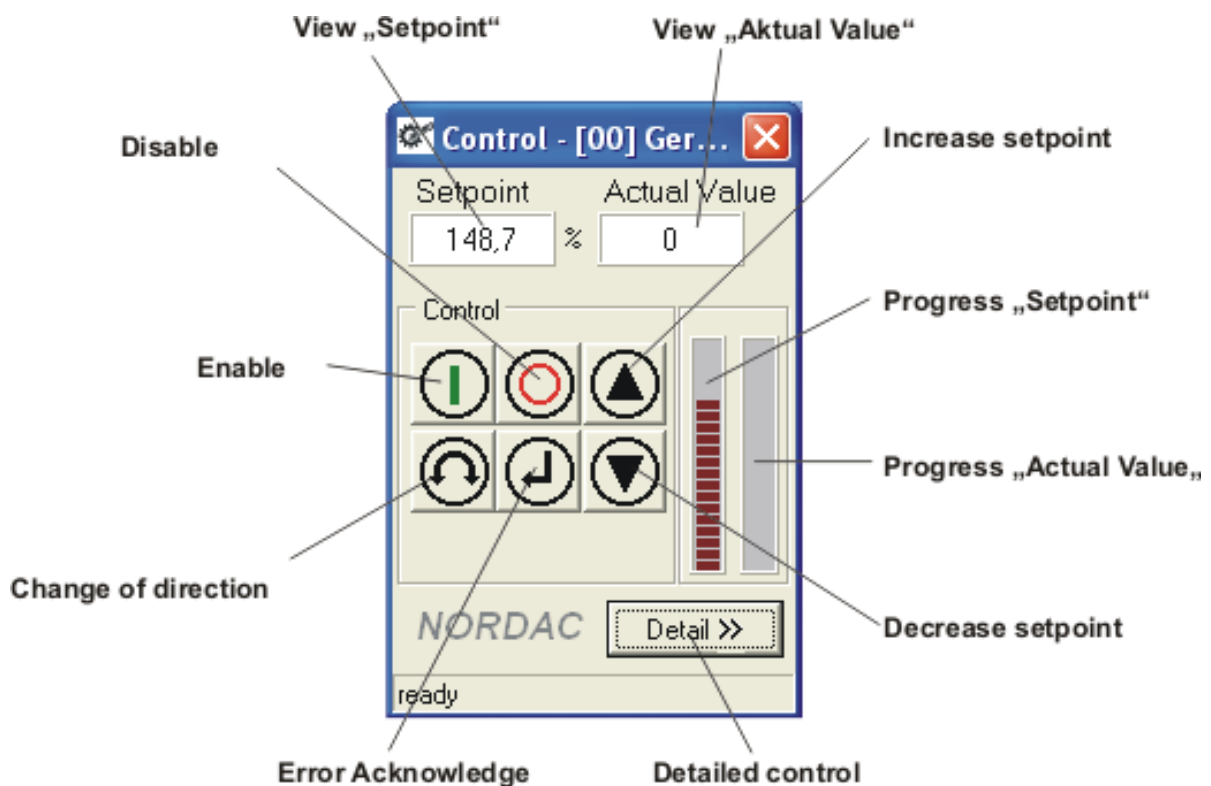
5.3.1 "Overview" All controls can be used with the window.

## 5.2 Standard control

Using the Standard control the following functions are available:

- Enabling the frequency inverter
- Increase or decrease of the setting value
- Change of direction
- Error Acknowledge

To use this functionality, the inverter must be programmed for control via bus. You can find the required parameter and settings in the manual available for each inverter type.



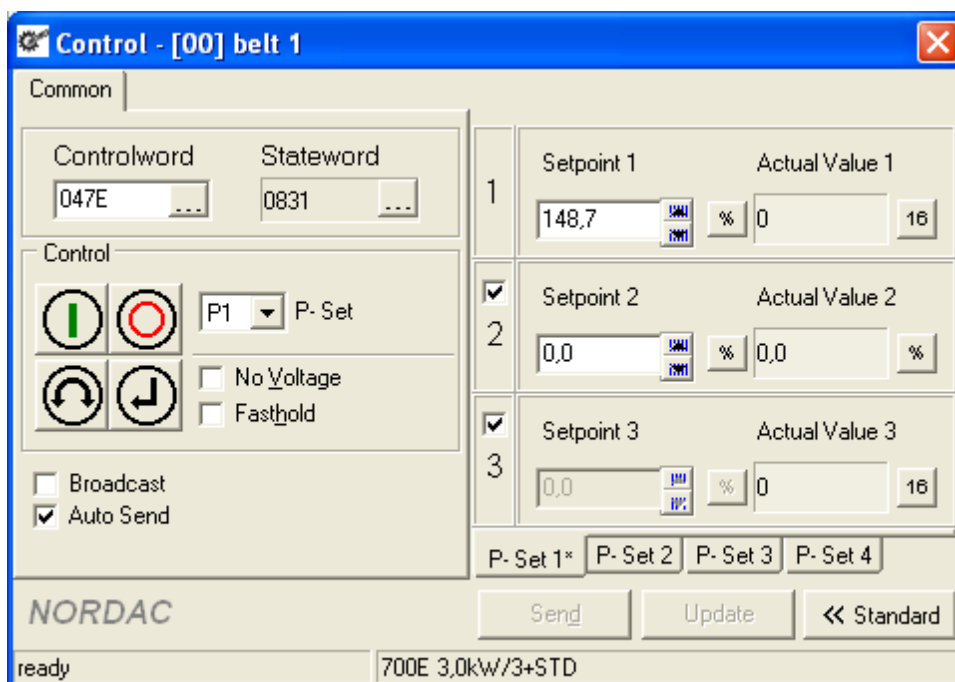
On the "Standard" display only the first setting value and first actual value are displayed. The form of value is fixed for each configuration. By pressing the button "Detail" you can switch to the extended control function.

## 5.3 Detailed control

### 5.3.1 Overview

In the mode „Detailed control" some extra functions are available:

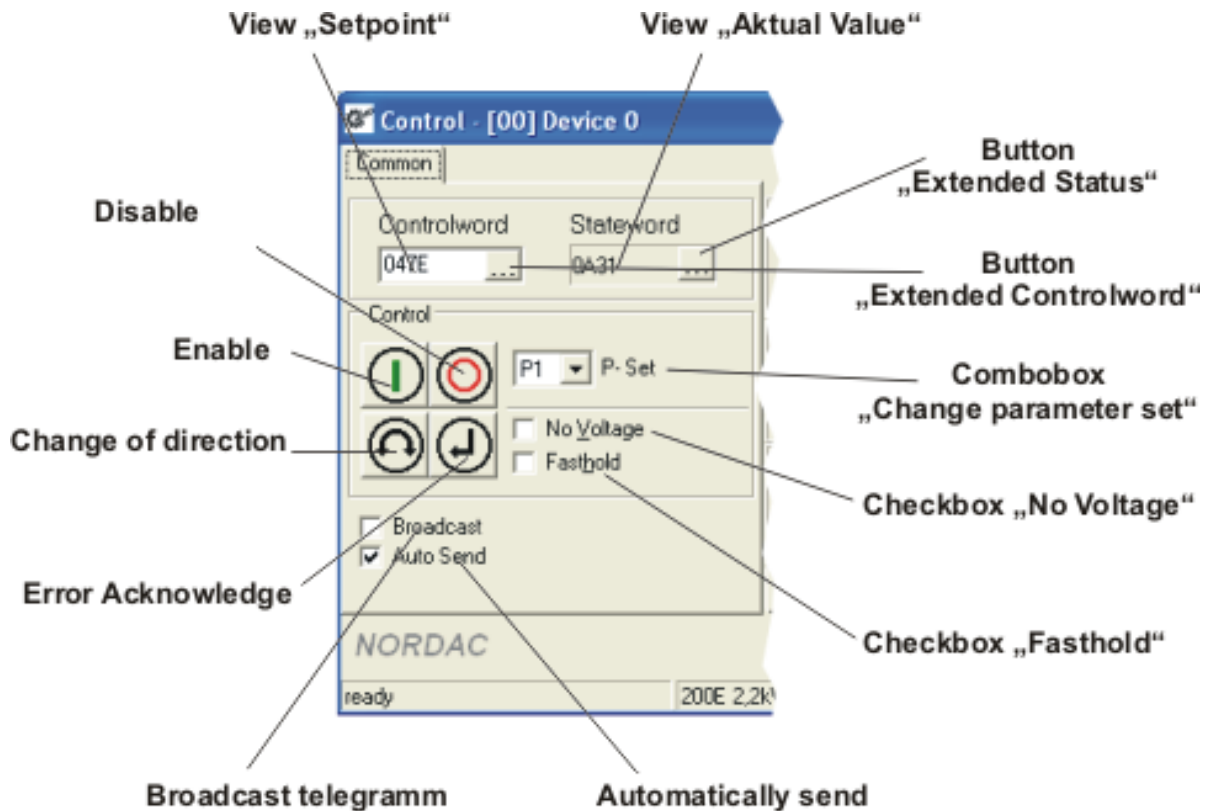
- 5.3.2 "Control "
- 5.3.3 "Management of setting values and actual values"
- Sending of broadcast telegram
- Choice of different parameter sets
- Automatic sending of control word and setting values



### 5.3.2 Control

The control word is displayed as a hexadecimal value in the field „Control word“. By entering a new value (hexadecimal) the user can change the control word. For a bit-coded setting of control word the user can open up a new editorial window by pressing the button "Control word edit". In this window the control word is displayed in bits.

The status word is displayed as a hexadecimal in the screen „Status word“. To display the status word in the bit resolution the button „Bit orientated detail view“ can be chosen. The status is displayed in the status line of the status machine as clear text.



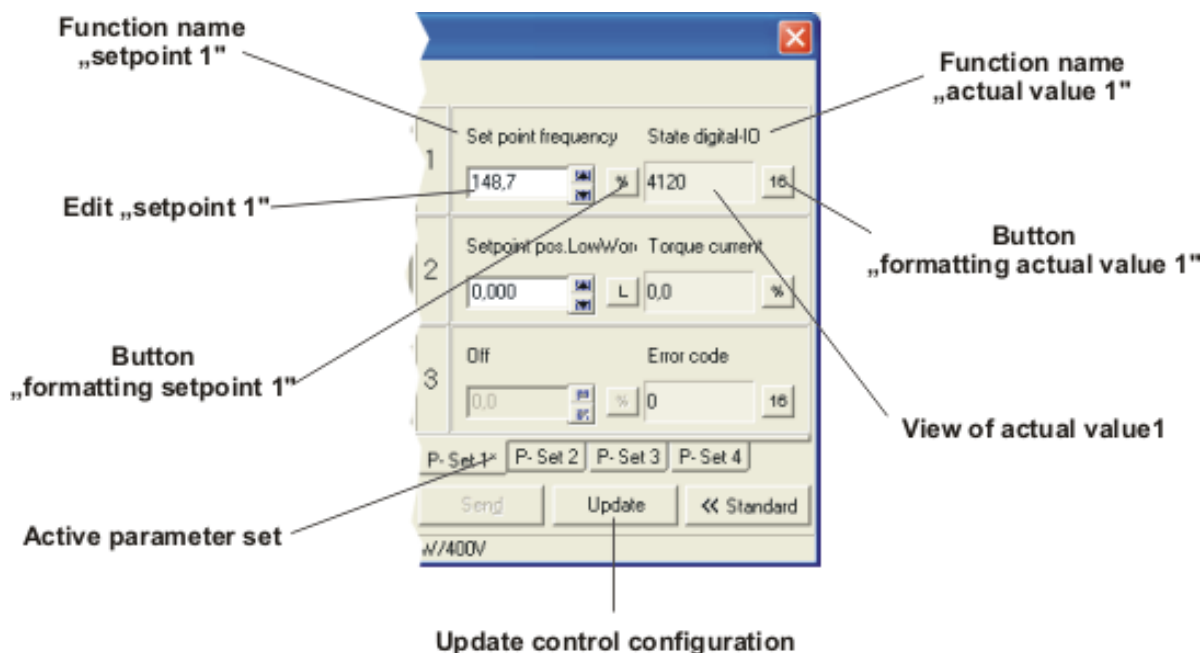
### 5.3.3 Management of setting values and actual values

For controlling the inverter the user can define up to 3 different setpoints and actual values (see user manual). The setpoints and actual values are displayed according to the **Fehler! Verweisquelle konnte nicht gefunden werden. "Fehler! Verweisquelle konnte nicht gefunden werden."** (Button „formatting setting value x“). The input of setpoints can be done in same way.

With the option „Setting/control/ parameter set individual management“ the setting values and actual values can be managed individually. So setting values can be set for each parameter set. With activation of a parameter set its setting values are transmitted to the frequency inverter. This is necessary because for each parameter set the setting values and actual values can be defined individually. The active parameter set is marked with a star.

If the option „Setting/control/configuration automatically checked“ was not activated, the user can transmit the new configuration by pressing the button „Update“.





### 5.3.4 Formatting of Setpoint and/or actual value




Char	Name	Description
"%"	16 Bit standardised values	This standardisation transforms the setpoint/actual value to a 16 Bit standardised value. Standardisation means a scaling of value range and is between -200% and 199% of a basic value (e.g. nominal frequency).
"16"	16 Bit not standardised	With this formatting the setpoint and actual value are transformed to 16 Bit value and transmitted to inverter and displayed without any scaling.
"B"	DigInBits	With this Formatting the setpoint and actual value are transformed to 8 Bit value. The bit status is displayed individually in check boxes. In these check boxes each bit of the setting value can be changed.
"L"	32 Bit Low-Word	With this formatting the setpoint and actual value are taken as the low word (16 Bit) of a 32 Bit word. If there is another setpoint or actual value parametrised with formatting "32 Bit High-Word", then both values are combined in the top display. The setting value can be given as a 32 Bit value.
"H"	32 Bit High-Word	With this formatting the setpoint and actual value are taken as the high word (16 Bit) of a 32 Bit word. (see "32 Bit Low-Word").

### 5.3.5 Status word

The present status word is displayed with each bit in the window „Status word“. All bits are listed in a table including bit number, name and status. According to bit value and function there is a coloured LED shown.


#### Importance of LEDs:

















LED	Importance
-----	------------

	The Bit is set and/or the inverter is enabled.
	An error is active or an enable signal is missing.
	The Bit is not set.

With the standard setting the status word is read in cycles and the changes are displayed in the window. For deactivating the cyclic reading switch off the function „Automatic" in the menu (right mouse click).

The window is docked left next to the „Control" window. If the window should be free on the desktop, the user should choose the popup menu "Docking/no". To save space the window can be added as an index card next to the index card "General". To do this the window must be moved (pressed left mouse button) over the index card "General". After release of the button the window is shown as an index card. With a double click (left mouse button) on the index card the user will get back to window mode.






Bit	Name	State
0	Ready to start	 1
1	Ready for operation	 0
2	Enabled	 0
3	Error	 0
4	Voltage enabled	 1
5	Fasthold	 1
6	No starting lockout	 0
7	Warning activ	 0
8	Setpoint reached	 1
9	Bus control active	 1
10	Start function 481.9	 1
11	Turn right on	 1
12	Turn left on	 0
13	Start function 481.10	 1
14	Parameterset bit 0 on	 0
15	Parameterset bit1 on	 0

### 5.3.6 Control word

The present control word is displayed with each bit in the window „Control word". All bits are listed in a table including bit number, name and status. According to bit value and function there is a coloured LED shown. If inverter is programmed to USS control then the bits can be set by control buttons. Each change of control word is sent immediately to the inverter (see „Automatic sending").

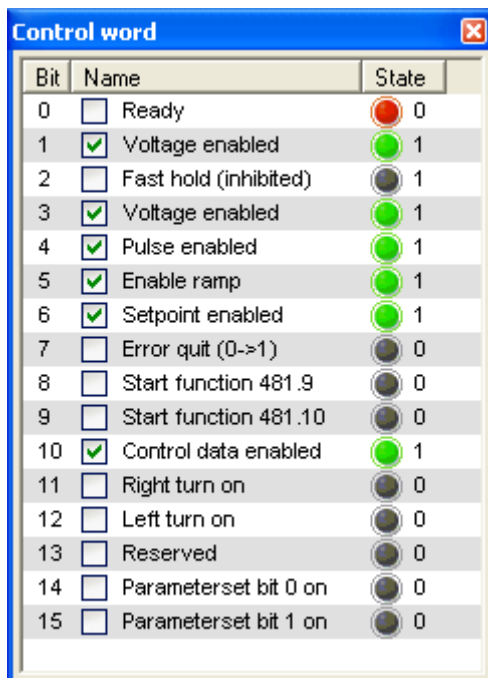
#### Importance of LEDs:
















LED	Importance
	The Bit is set and/or the inverter is enabled.

	An error is active or an enable signal is missing.
	The Bit is not set.

With the standard setting the control word is read in cycles and the changes are displayed in the window. For deactivating the cyclic reading switch off the function „Automatic" in the menu (right mouse click).

The window is docked left next to the „Control" window. If the window should be free on the desktop, you should choose the popup menu "Docking/no". To save space the window can be added as an index card next to the index card "General". To do this the window must be moved (pressed left mouse button) over the index card "General". After release of the button the window is shown as an index card. With a double click (left mouse button) on the index card the user will get back to window mode.



Bit	Name	State
0	<input type="checkbox"/> Ready	 0
1	<input checked="" type="checkbox"/> Voltage enabled	 1
2	<input type="checkbox"/> Fast hold (inhibited)	 1
3	<input checked="" type="checkbox"/> Voltage enabled	 1
4	<input checked="" type="checkbox"/> Pulse enabled	 1
5	<input checked="" type="checkbox"/> Enable ramp	 1
6	<input checked="" type="checkbox"/> Setpoint enabled	 1
7	<input type="checkbox"/> Error quit (0->1)	 0
8	<input type="checkbox"/> Start function 481.9	 0
9	<input type="checkbox"/> Start function 481.10	 0
10	<input checked="" type="checkbox"/> Control data enabled	 1
11	<input type="checkbox"/> Right turn on	 0
12	<input type="checkbox"/> Left turn on	 0
13	<input type="checkbox"/> Reserved	 0
14	<input type="checkbox"/> Parameterset bit 0 on	 0
15	<input type="checkbox"/> Parameterset bit 1 on	 0

## 6 Remote

NORD CON can simulate the control unit of the respective frequency inverter. For this purpose the frequency inverter transfers the content of its display to NORD CON. The key functions are simulated on the PC and transmitted to the frequency inverter.

The frequency inverter can only be controlled via the Remote, if it has not previously been enabled via the control terminals or via a serial interface (P509 = 0 and P510 = 0). In addition, for this the parameter "PotentiometerBox Function" (P549) must not be set to function {4} "Frequency addition" or function {5} "Frequency subtraction".

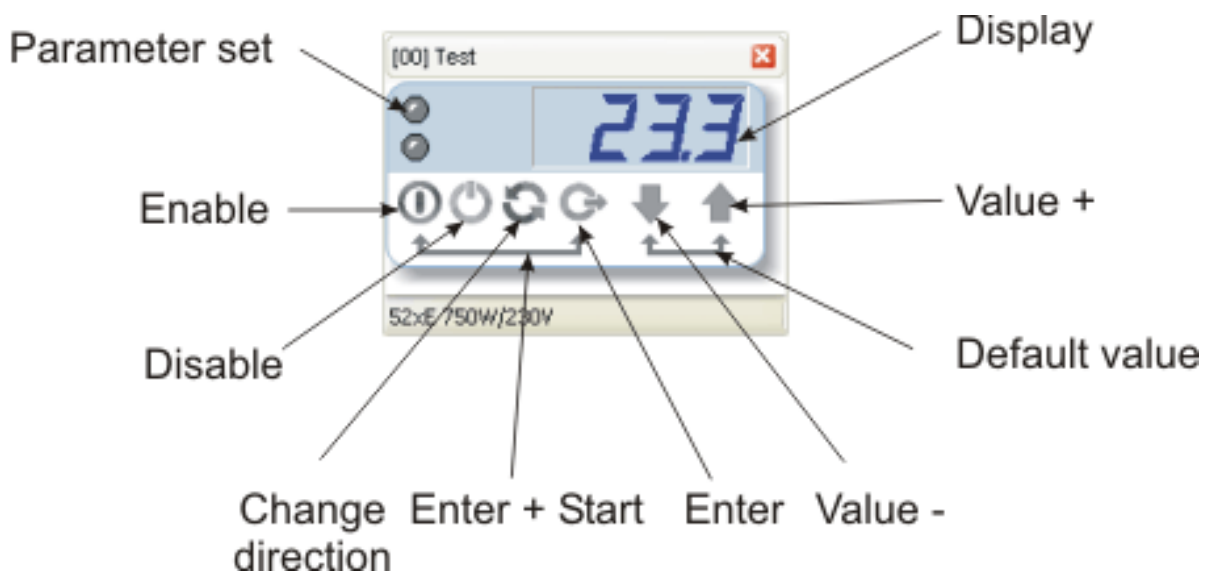
### **i** Information




### Timeout monitoring function

NORD Frequency inverter können über die Tastatur gesteuert werden (Freigabe, Sollwert +/-, Drehrichtung, etc.). Dabei ist die Time-Out-Überwachung nicht aktiv, sodass bei Abbruch der Verbindung zwischen PC und Frequenzumrichter kein Steuern mehr möglich ist.

### 6.1 Standard

The standard window for the function "Remote" is used for all Devices, if the option "13.1 "User interface"" is not activated.



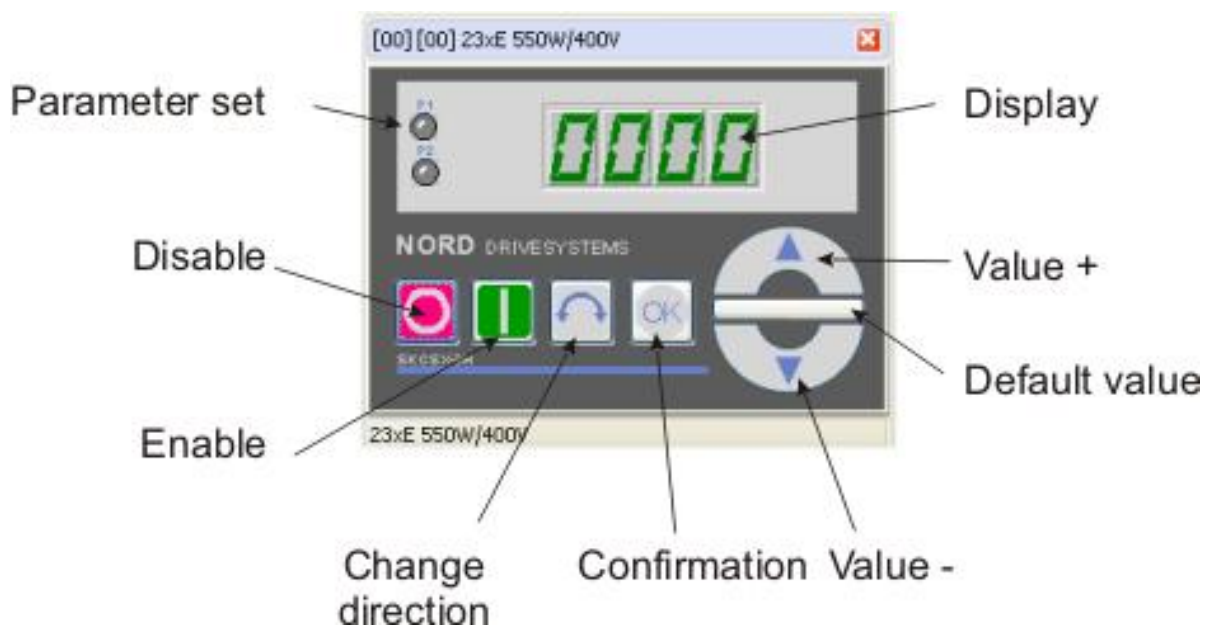
Name	Icon	Description
Enable		Switching on the frequency inverter. The frequency inverter is now enabled with the set jog frequency (P113). A preset minimum frequency (P104) must at least be provided. Parameter >Interface< P509 and P510 must = 0.
Disable		Switching off the frequency inverter. The output frequency is reduced to the absolute minimum frequency (P505) and the frequency inverter shuts down.
Change dir		The motor rotation direction changes when this key is pressed. "Rotation to the left" is indicated by a minus sign.  <b>Attention:</b>

Name	Icon	Description
		Take care when operating pumps, screw conveyors, ventilators, etc. Block the key with parameter P540.
Up	↑	Press key to increase the frequency. During parameterisation, the parameter number or parameter value is increased.
Down	↓	Press the key to reduce the frequency. During parameterisation, the parameter number or parameter value is reduced.
Enter	↻	Press "ENTER" to store an altered parameter value, or to switch between parameter number or parameter value.  <b>Note:</b> If a changed value is not to be stored, the key can be used to exit the parameter without storing the change.
Change Dir + Stop		By simultaneously pressing the STOP key and the "Change direction key", a quick stop can be initiated.
Enter + Start		If the inverter is enabled via the "ON" key, the parameterisation mode can be reached by pressing the ON and ENTER keys simultaneously.






All functions available with the operating unit (control box) of the frequency inverter can be performed.

## 6.2 NORDAC SK 200 E

The window for remote control of the frequency inverters of the NORDAC SK 200 E series looks like this:



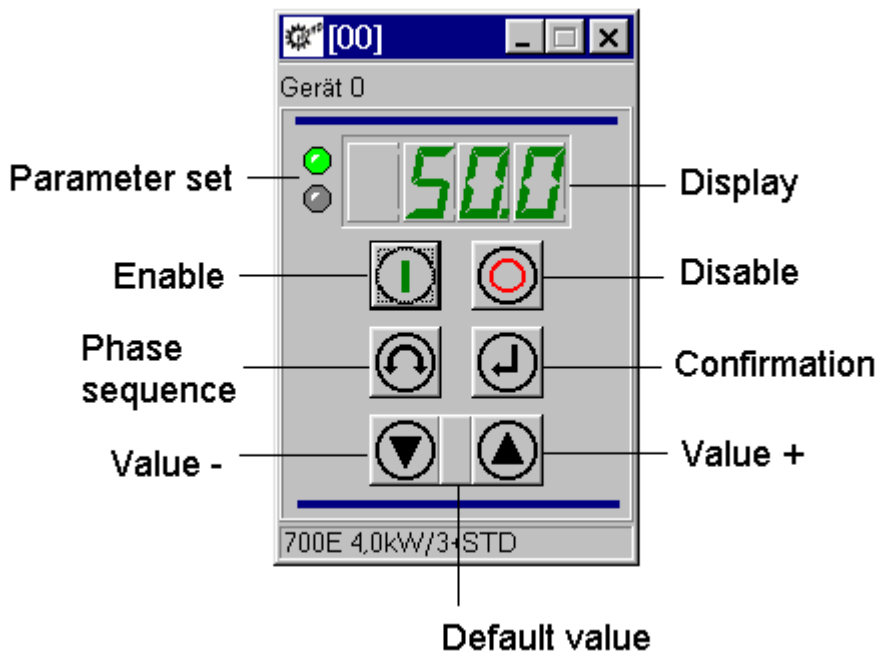
Name	Icon	Description
Enable	ⓘ	Switching on the frequency inverter. The frequency inverter is now enabled with the set jog frequency (P113). A preset minimum frequency (P104) must at least be provided. Parameter >Interface< P509 and P510 must = 0.







Name	Icon	Description
Disable		Switching off the frequency inverter. The output frequency is reduced to the absolute minimum frequency (P505) and the frequency inverter shuts down.
Change dir		The motor rotation direction changes when this key is pressed. "Rotation to the left" is indicated by a minus sign.  <b>Attention:</b> Take care when operating pumps, screw conveyors, ventilators, etc. Block the key with parameter P540.
Up		Press key to increase the frequency. During parameterisation, the parameter number or parameter value is increased.
Down		Press the key to reduce the frequency. During parameterisation, the parameter number or parameter value is reduced.
Enter		Press "ENTER" to store an altered parameter value, or to switch between parameter number or parameter value.  <b>Note:</b> If a changed value is not to be stored, the key can be used to exit the parameter without storing the change.
Change Dir + Stop		By simultaneously pressing the STOP key and the "Change direction key", a quick stop can be initiated.
Enter + On		If the inverter is enabled via the "ON" key, the parameterisation mode can be reached by pressing the ON and ENTER keys simultaneously.

All functions available with the operating unit (control box) of the frequency inverter can be performed.

### 6.3 NORDAC SK 700/500/300 E

The window for remote control of the frequency inverters of the NORDAC SK 700/500/300 E series looks like this:

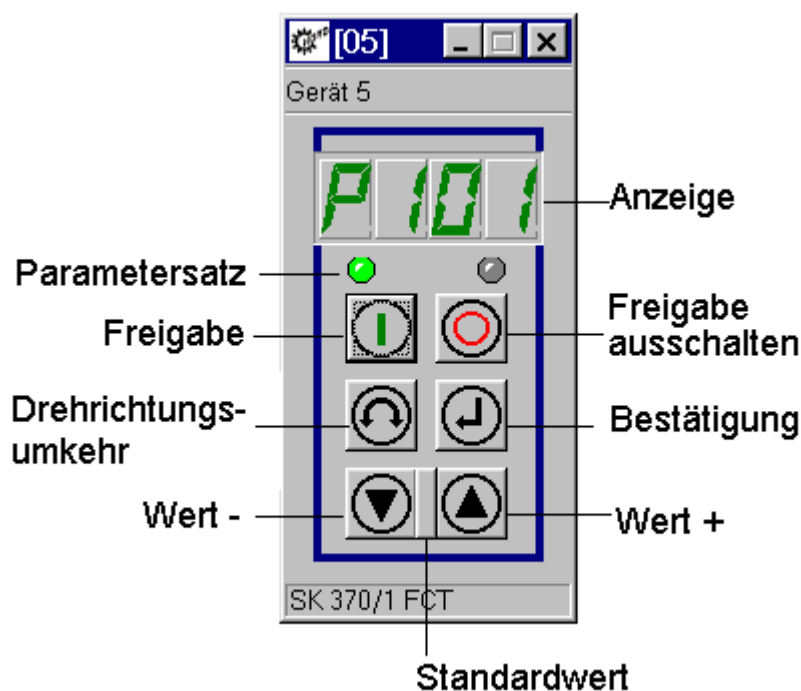








Name	Icon	Description
Enable		Switching on the frequency inverter. The frequency inverter is now enabled with the set jog frequency (P113). A preset minimum frequency (P104) must at least be provided. Parameter >Interface< P509 and P510 must = 0.
Disable		Switching off the frequency inverter. The output frequency is reduced to the absolute minimum frequency (P505) and the frequency inverter shuts down.
Change dir		The motor rotation direction changes when this key is pressed. "Rotation to the left" is indicated by a minus sign.  <b>Attention:</b> Take care when operating pumps, screw conveyors, ventilators, etc. Block the key with parameter P540.
Up		Press key to increase the frequency. During parameterisation, the parameter number or parameter value is increased.
Down		Press the key to reduce the frequency. During parameterisation, the parameter number or parameter value is reduced.
Enter		Press "ENTER" to store an altered parameter value, or to switch between parameter number or parameter value.  <b>Note:</b> If a changed value is not to be stored, the key can be used to exit the parameter without storing the change.
Change Dir + Stop		By simultaneously pressing the STOP key and the "Change direction key", a quick stop can be initiated.
Enter + On		If the inverter is enabled via the "ON" key, the parameterisation mode can be reached by pressing the ON and ENTER keys simultaneously.

All functions available with the operating unit (control box) of the frequency inverter can be performed.

### 6.4 NORDAC vector mc

The window for remote control of the frequency inverters of the NORDAC vector mc series looks like this:



Name	Icon	Description
Enable		Switching on the frequency inverter. The frequency inverter is now enabled with the set jog frequency (P113). A preset minimum frequency (P104) may at least be provided. Parameter >Interface< P509 and P510 must = 0.
Disable		Switching off the frequency inverter. The output frequency is reduced to the absolute minimum frequency (P505) and the frequency inverter shuts down.
Change dir		The motor rotation direction changes when this key is pressed. "Rotation to the left" is indicated by a minus sign.  <b>Attention:</b> Take care when operating pumps, screw conveyors, ventilators, etc. Block the key with parameter P540.
Up		Press key to increase the frequency. During parameterisation, the parameter number or parameter value is increased.
Down		Press the key to reduce the frequency. During parameterisation, the parameter number or parameter value is reduced.
Enter		Press "ENTER" to store an altered parameter value, or to switch between parameter number or parameter value.

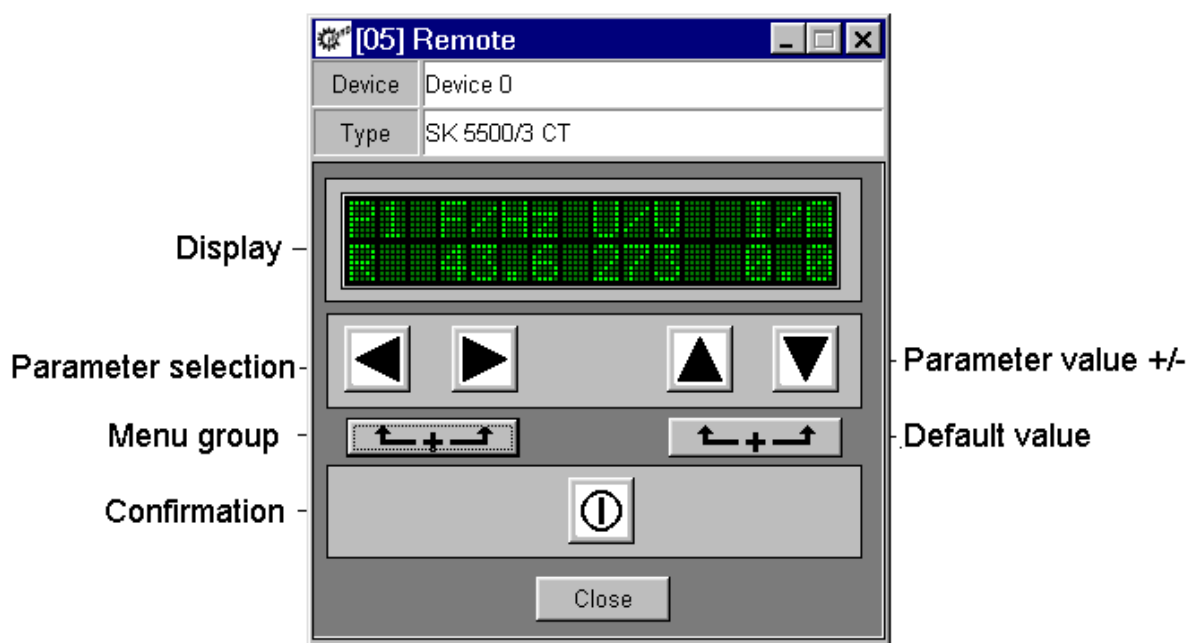





Name	Icon	Description
		<b>Note:</b> If a changed value is not to be stored, the key can be used to exit the parameter without storing the change.
Change Dir + Stop		By simultaneously pressing the STOP key and the "Change direction key" , an quick stop can be initiated.
Enter + On		If the inverter is enabled via the "ON" key, the parameterisation mode can be reached by pressing the ON and ENTER keys simultaneously.




All functions available with the operating unit (control box) of the frequency inverter can be performed.

## 6.5 NORDAC vector ct

The remote control window for the NORDAC vector ct series has the following appearance:



Name of action	Picture	Description
Enable		To switch on the frequency inverter. The frequency inverter is now enabled with the set jog frequency (P113). A pre-set minimum frequency (P104) may at least be provided. Parameter >Interface< P509 and P510 must = 0.
Switch off enable		To switch on the frequency inverter. The frequency inverter is now enabled with the set jog frequency (P113). A reset minimum frequency (P104) may at least be provided. Parameter >Interface< P509 and P510 must = 0.
Change direction of rotation		The direction of rotation of the motor changes when this key is pressed. "Rotation to the left" is indicated by a minus sign.  <b>Notice:</b> Take care when operating pumps, screw conveyors, ventilators, etc. Block the key with parameter P540.

Name of action	Pictu re	Description
Increase		<p>The direction of rotation of the motor changes when this key is pressed. "Rotation to the left" is indicated by a minus sign.</p> <p><b>Notice:</b> Take care when operating pumps, screw conveyors, ventilators, etc. Block the key with parameter P540.</p>
Reduce		<p>The direction of rotation of the motor changes when this key is pressed. "Rotation to the left" is indicated by a minus sign.</p> <p><b>Notice:</b> Take care when operating pumps, screw conveyors, ventilators, etc. Block the key with parameter P540.</p>
Confirm		<p>Press this key to store a changed parameter value, or to switch between the parameter number and the parameter value.</p> <p><b>Note:</b> If a changed value is not to be stored, the key can be used to exit from the parameter without saving the change.</p>
Rotation direction + Switch off enable		By simultaneously pressing the STOP key and the "Change direction key ", a quick stop can be initiated.
Confirm + enable		Simultaneously pressing the ON key and the "Confirm" key switches to the editing mode for an enabled device.

All of the functions which are possible with the control unit (Control Box) can be carried out.

## 7 Oscilloscope

### 7.1 Overview

The oscilloscope function integrated in NORD CON can show process data of an NORD Frequency inverter as an arithmetic chart.

#### **i** Information

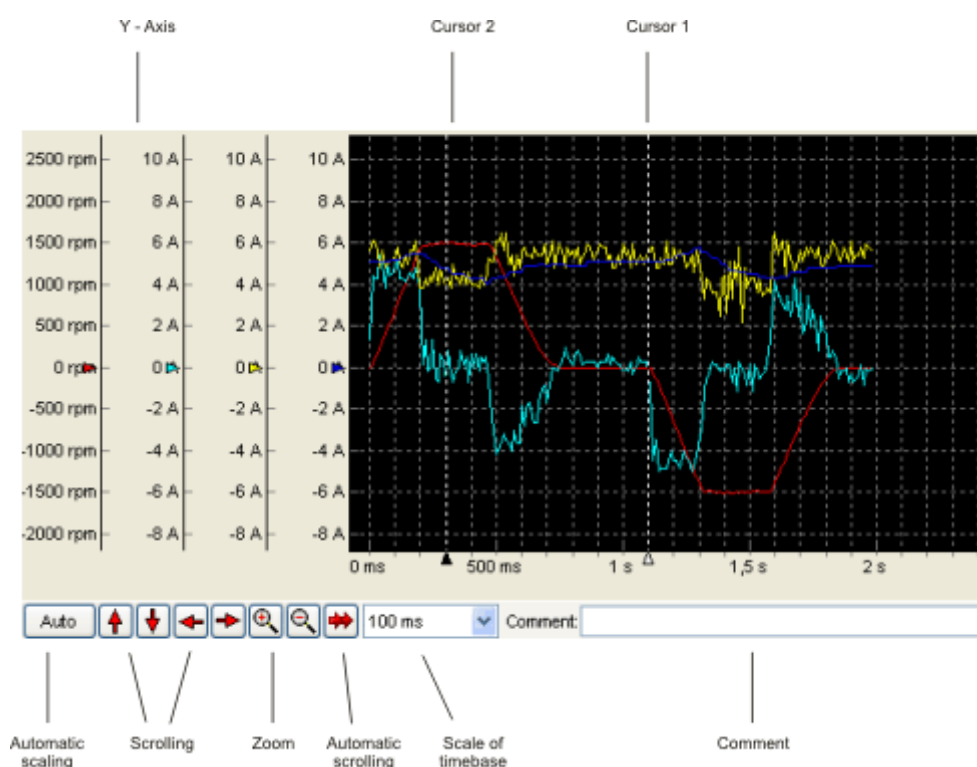
This option is not available for serieses NORDAC vector ct and NORDAC vector mc!

The features of oscilloscope-function are:

- Monitoring of up to 4 channels
- Many different ways of triggering
- Scaling of each measurement
- Calculation of average values, effective value, etc.
- Save, print and export of measurement data

### 7.2 Display

The oscilloscope function can measure and display 4 channels max:



The following settings can be done:

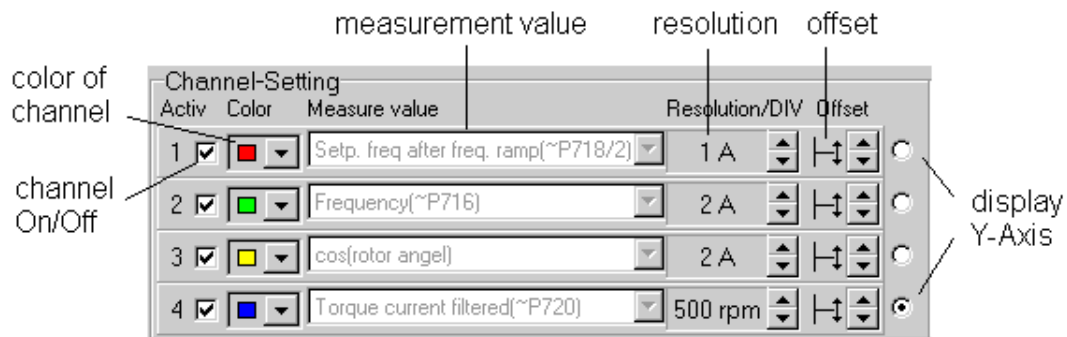
Name	Description
------	-------------

Auto	Automatic scaling of all measured data
Offset	Selection of display detail (displacement of all data in x- or y-direction)
Zoom	Display size (Zoom of all data)  <b>Note:</b> With the right mouse button you can choose between the modes "Move" and "Measurement", if the mouse pointer is on the display. In "Move" mode you can choose the detail of display by mouse pointer by pressing the left mouse button while moving over the display.
Auto scrolling	With this option during a recording the time axis is scrolled automatically to the last point.
Resolution	In this combination field the user can change the scaling of the time axis.
Comment	Additional information field, where further information for the measurement series can be stored.
Cursor	Execution of measurement

### 7.3 Handling

Follow the next steps to execute a measurement:

#### 1. Choice of channels



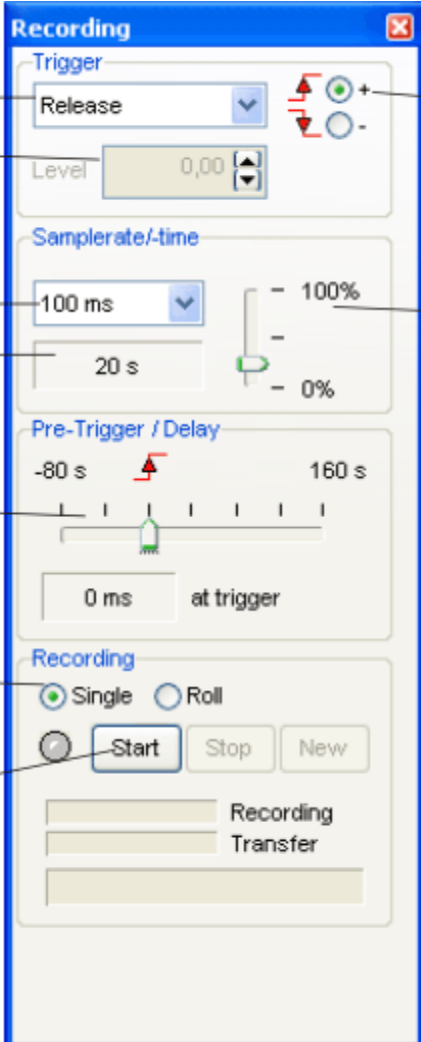
There is a popup menu to make the choice of the 4 channels. There is a colour referring to each channel. Each channel can be switched on and off by checkboxes. The resolution and offset can be chosen for each channel separately. When displaying the results of measurement, the values of the vertical axis of each channel can be chosen and indicated.

#### Importance of measured value

Measured value	Description
(=P[Number]) [Name]	The value of this measuring function is updated in the time slot pattern by approx. 100 milliseconds and corresponds to the value indicated of the parameter.
[Name]	The value of this measuring function is updated in a time slot pattern by approx. 100 milliseconds.

(≈P[Number]) [Name]	The value of this measuring function is updated in a time slot pattern by approx. 50 milliseconds.
(~P[Number]) [Name]	The value of this measuring function is updated in a time slot pattern of approx. 250 μs.

## 2. Setting of trigger



The screenshot shows the 'Recording' configuration window with the following settings and labels:

- Source of trigger:** Release (dropdown menu)
- Trigger level:** 0,00 (input field)
- Trigger edge:** + (radio button selected), - (radio button)
- Sampling rate:** 100 ms (dropdown menu)
- Time of sampling:** 20 s (input field)
- Number of measured values:** 100% (slider)
- Pre-Trigger / Delay:** -80 s to 160 s (range), 0 ms at trigger (input field)
- Sampling modes:** Single (radio button selected), Roll (radio button)
- Start/Stop sampling:** Start, Stop, New (buttons)

The trigger starts the measurement. First choose the source of trigger. Trigger sources can be measurement values, digital inputs, status of inverter, etc. The starting conditions are defined by trigger level respectively trigger edge.

### **i** Information

#### Trigger levels

The increments of the trigger levels are different depending on the trigger source. Therefore not every value can be set. After starting a recording, the closest possible value is calculated and set.

Time between two measured values is set by sampling rate. Numbers of measured values and sampling rate define the time of sampling. The Pre-trigger/Delay set the beginning of the measurement in relation to the trigger event.

**i Information**

**Measured values**

The dynamic of measured values defines the best rate of sampling: fast changing values need a low sampling rate. The number of measured values defines the time of sending the values from inverter to NORD CON.

**3. Sampling modes**

The oscilloscope has 2 different modes. The user can choose between "Single" and "Roll" mode. The "Single" mode is the standard mode. In this mode a recording starts with the current trigger settings. The recording time depends on the oscilloscope memory of the device and amounts to max. 2000 seconds. The values are noted in the adjusted sampling rate.

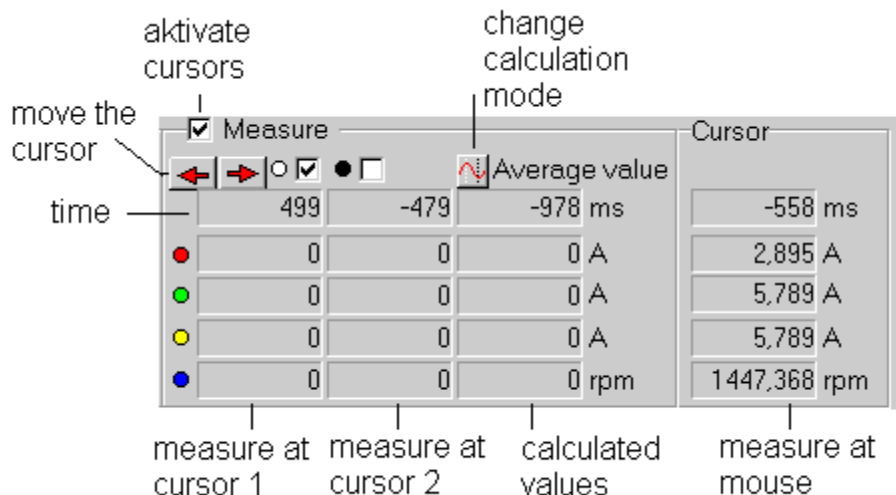
The roll mode makes a recording over longer period. The noted values are transferred immediately to the PC. Therefore the user cannot change the sampling rate. It depends on the speed of the transmission.


**4. Starting of measurement**

The Start-Button activates the measurement. The event of trigger is detected. When the event appears a recording starts in the inverter. The transmission of data to NORD CON starts in the same moment. This can be cancelled by Stop. After transferring all data a new measurement can be started or new settings can be made by pressing the "New" button.

**7.4 Measurement**

After recording the measurement completely, measurements on the results can be done using cursors.



There are two cursors available for this. The cursors can be moved by . The choice of cursor is made by    . To choose the mode "Move" and "Measurement" by right mouse button the pointer has to be on display. In the measure mode the cursors can be set by left mouse button. The values of the measured lines 1 and 2 are displayed on cursor 1 and cursor 2. Additionally the calculations like average values are performed. Pressing on the "Calculation" button starts the shift of calculation.

## 7.5 Save and Print

The recorded series of measurements can be saved, exported or printed.

### Menu item "File"

Name	Description
Open	A stored measurement data file can be chosen and loaded. During loading there is a choice if only the setting should be loaded or all data of measurement.
Save as	The present measurement data and settings are saved with a new filename.
Export	The data can be exported as graphic file or data table.
Print	The lines of measurement are printed with present settings (colour of background: white).

### Scope Offline

In Offline-mode (no inverter is connected) a saved measurement file can be loaded by menu item File|Open.

## 8 Macro editor

The macro editor is designed to create simple process sequences. The user interface provides a facility for creating and adapting a macro using context menus, toolbars or tool windows. The individual instructions can be moved within the view using Drag n Drop. The standard functions such as saving and loading a macro are also integrated in the context menu. The macros are stored in the standard format "XML". The format of the preceding version can be imported using the "Open" menu item, file type "Macro Files V1.26".

### 8.1 User interfaces and views

As well as the editor window, other views are also needed for macro generator handling. These views are available as tool windows. These windows can be docked and undocked at the edge of the main window. All views can be displayed and closed using the "View" menu item in the pop-up menu.

#### 8.1.1 Window "Variables"

The view „variables" can be opened and closed over the menu option „View->Macro->Variables ". It is used for debugging. In this window after starting macros all variables and objects macros with current rating are indicated. The expenditure of the value can be stopped in the view „Properties->Display format".

**There are the following formatting:**

- Decimal
- Hexadecimal
- Binary

#### 8.1.2 Properties window

The "Property" view can be opened and closed via the "View -> Properties" menu item. All properties of the current instruction are displayed in this window. Depending on the instruction, the number of properties and the type thereof can change.

Name	Description
Result	You can change the object to which you would like to assign a new value with this property. Only objects to which a new value can be assigned can be selected (e.g. control word, parameters or variables).
Operand	With this property the user can select the object that is to be used with an assignment or operation.
Operator	This property defines the type of operation (e.g. Addition).
Comment	The user can assign a comment to any instruction using this property.

Variables, control or status words, setpoints or actual values or parameters can be designated as objects in the macro generator. Each of these objects has different parameters.



Object	Parameter	Description
Variable	Name	The parameter defines the name of the variable or constant. All variables that have already been used are displayed in the selection box. If you would like to create a new variable, a name that has not yet been used must be entered. No distinction is made between upper and lower case.
	Display format	This parameter defines the display format in the “Variables” view. One of the following displays can be selected: <ul style="list-style-type: none"> <li>• Decimal</li> <li>• Hexadecimal</li> <li>• Binary</li> </ul>
Constant	Value	The parameter defines the value of the constant.
	Display format	This parameter defines the display format in the “Variables” view. One of the following displays can be selected: <ul style="list-style-type: none"> <li>• Decimal</li> <li>• Hexadecimal</li> <li>• Binary</li> </ul>
Control word, Status word	Node number	This parameter defines the USS node number of the required device.  <b>Note:</b> Since the current control word cannot be read out of the device, the control word is set to 0 when the scheduler starts.
	Display format	This parameter defines the display format in the “Variables” view. One of the following displays can be selected: <ul style="list-style-type: none"> <li>• Decimal</li> <li>• Hexadecimal</li> <li>• Binary</li> </ul>
Setpoint and actual values	Node number	This parameter defines the USS node number of the required device.  <b>Note:</b> Since the current setpoints cannot be read out of the device, the values are set to 0 when the scheduler starts.
	Type	This parameter defines the type of the value. The types listed in table “Setpoint and actual value types” are available to the user.
	Format	This parameter defines the formatting of the setpoint and actual values. The possible formats are shown in table “Setpoint and actual value formatting”.
	Resolution	This parameter defines the resolution of the setpoint and actual values. It is used to format the instruction in the editor.
	Display format	This parameter defines the display format in the “Variables” view. One of the following displays can be selected: <ul style="list-style-type: none"> <li>• Decimal</li> <li>• Hexadecimal</li> <li>• Binary</li> </ul>
Parameter	Node number	This parameter defines the USS node number of the required device.

Object	Parameter	Description
	Parameter number	This value defines the number of the parameter (see "Device catalogue" view).
	Sub-index	This value defines the sub-index of the parameter.
	Resolution	This value defines the resolution of the setpoint and actual values. It is used to format the instruction in the editor.
	Data type	This value defines the data type of the parameter. Only 2 data types are used in the current devices (16-bit integer and 32-bit integer).
	Display format	This parameter defines the display format in the "Variables" view. One of the following displays can be selected: <ul style="list-style-type: none"> <li>• Decimal</li> <li>• Hexadecimal</li> <li>• Binary</li> </ul>

### Setpoint and actual value types

Type	Description
Value 1 (16-bit)	The 1st, 2nd or 3rd setpoint or actual value should be used.
Value 12 (32-bit)	The first and second setpoint or actual value should be used as a 32-bit value.  <b>Note:</b> The device must be appropriately configured for this configuration (see "Setpoint or actual value configuration").
Value 13 (32-bit)	The 1st and 3rd setpoint or actual value should be used as a 32-bit value.  <b>Note:</b> The device must be appropriately configured for this configuration (see "Setpoint or actual value formatting").
Value 23 (32-bit)	The 2nd and 3rd setpoint or actual value should be used as a 32-bit value.  <b>Note:</b> The device must be appropriately configured for this configuration (see "Setpoint or actual value formatting").

### Setpoint and actual value formatting

Formatting	Description
Standardised	This formatting interprets the setpoint or actual value as a 16 bit standardised value. Standardisation means scaling of the value range and is between -200% and 199% of a basic value (e.g. nominal frequency).
Not standardised	In this formatting the setpoint or actual value is interpreted as a 16 bit value, which is transferred to the device and displayed without scaling.

Formatting	Description
Low word (32-bit)	This formatting defines that the first word is the low word and the 2nd value is the high word value (32-bit). This value can only be selected for 32-bit types.
High word (32-bit)	This formatting defines that the first word is the high word and the 2nd value is the low word value (32-bit). This value can only be selected for 32-bit types.

**Note:**

Please ensure that the configuration of the devices corresponds with the settings.

### 8.1.3 Log window

All events in the sequence control are saved in a log. To display the log, you need to open the “Log” view using the “View -> Log” menu item. The value is also a tool window and can be docked and undocked at the edge of the main window. All log entries are shown in a sorted list in the window. In this case, the last entry is at the beginning of the list.

#### Saving the log

The log can be saved using the “Save as...” menu item in the pop-up menu. A file selection dialogue then opens, and the user must stipulate the name and path for storing the log file. If the user confirms with “Save”, the current list is saved in the text file.

#### Deleting the log

The log can be deleted using the “Delete” menu item in the pop-up menu. All entries are then irretrievably deleted.

#### Filtering the entries

The user can filter the log entries in accordance with the type of entry using the filter function. The types of the entries to be entered in the log can be defined using the “Filter” menu item.

## 8.2 Working with macros

### 8.2.1 Create a new macro

A new document (macro) is generated by the menu option “New” in the context menu. Depending if the document was previously opened, the macro editor offers to store of the old document. A new document is generated if the user does not confirm with “Cancel”. Only one document can be opened at the same time in the current version.

### 8.2.2 Open a macro

Opening macros is implemented in the menu option “Open” or with the combination of keys “Ctrl+O”. Subsequently, a selection of files dialogue opens, in which the user can select the desired macro. If the user would like to open a previous version of the macro, he must change the data type in the file selection dialogue accordingly.

### 8.2.3 Save a macro

Storing macros is implemented in the menu option "Save" or the combination of keys a "Ctrl+S". This function is available however only for previously generated documents. For all new documents the function must be implemented "Save as...".

The function is implemented in the menu option "Save as...". Subsequently, a selection of files dialogue opens, in which the user must select the file name as well as the path. After confirmation with "Save" the macro is stored. After the completion of the procedure the newly named macro indicated in the title bar.

### 8.2.4 Inserting instructions

The "Insert" function is activated using the "Insert" menu item or key combination "Ctrl+V". It inserts a previously copied or cut instruction below the current position in the document. If no instruction has been copied or cut beforehand, the menu item is deactivated. In the current version, each copied or cut instruction can only be inserted once.

### 8.2.5 Copying instructions

The "Copy" function is activated using the "Copy" menu item or key combination "Ctrl+C". It copies the selected line into the clipboard of the generator. Only one line can be selected in the current version. This means that only one instruction can be copied. The Block instruction is an exception. This can only be copied in its entirety.

### 8.2.6 Cutting instructions

The "Cut" function is activated using the "Cut" menu item or key combination "Ctrl+X". It copies the selected instruction into the clipboard of the generator. When the cut instruction is inserted, the old instruction is deleted from the document. The restriction that only one instruction can be cut also applies to this function.

### 8.2.7 Delete from instruction

The function is implemented in the menu option "Delete" or the combination of keys "Ctrl + Del". It deletes the marked instruction from the document.

### 8.2.8 Search and replace

The function "Search and replace" is implemented in the "Search and replace" menu or the combination of keys "Ctrl+H" where the dialogue "Search and replace" opens. This allows the user to insert the search and replacement vocabulary and start the change procedure.

### 8.2.9 Shift up instruction

The function is implemented in the menu option "Shift up". It shifts the marked instruction a line upward. If the top line of document is marked no action is implemented. Shifting of instructions can also be done by drag and drop with the mouse.

### 8.2.10 Shift down instruction

The function is implemented in the menu option "Down". It shifts the marked instruction one line downwards. If the last line of the document is marked no action is implemented. Shifting instructions can also be done by drag and drop with the mouse.

### 8.2.11 Creating new instructions

New instructions are created using the “Functions” menu item in the context menu. The new instructions are always inserted below the selected line. The user can then change the position of the new instruction (see “Up” and “Down”).

The following functions are available to the user in this version:

Name	Description
Assignment	<p>The instruction assigns a new value to a macro object. The new value can be read out of another object, or the user defines a constant. By default, the line is inserted as shown in example 1. The Parameter function can be adapted in the “Properties” view.</p> <p><b>Example:</b>            Device 00 Controlword = 047F hex // Assign value of 1151 to control word            Var1 = Device 00 Statusword // Assign value of status word to variable</p> <p><b>Note:</b>            Setpoints can only be assigned within a Block instruction.</p>
Jump mark	<p>The instruction defines a jumping point in the macro. The user can jump to the location of the jump mark using the “Goto” function. By default, the line is inserted as shown in example 1. The Parameter function can be adapted in the “Properties” view. The name of the jumping point must be changed, since duplicate names are not supported. The generator always jumps to the first jump mark in the macro.</p> <p><b>Example:</b>            Label1: // Defines the “Label1” jump mark.</p> <p><b>or</b>            Start: // Defines the “Start” jump mark</p>
Sleep	<p>The instruction generates a pause in the execution of the macro. The time is specified in units of “ms”. By default, the line is inserted as shown in example 1. The time can be adapted in the “Properties” view.</p> <p><b>Example:</b>            Sleep 1000 ms // Wait for 1s</p> <p><b>or</b>            Sleep 500 ms // Wait for 0.5s</p>
Go to	<p>The instruction generates a jump in the macro. After executing the instruction, the generator jumps to the line of the jump mark with the relevant name. If the generator does not find a jump mark with the name, the line is ignored. If no jump mark has been defined in the macro yet, the menu item is deactivated. The first jump mark is always entered by default. The name of the jump mark can be adapted in the “Properties” view.</p> <p><b>Example:</b>            Goto Start // Go to jump mark “Start”</p>
Condition	<p>The instruction generates a conditional jump in the macro. If the condition is true, the generator jumps to the line of the jump mark with the relevant name. By default, the line is inserted as shown in example 1. The parameters of the instruction can be adapted in the “Properties” view.</p> <p><b>Example:</b>            if Device 00 Controlword == 047F hex then // If the control word has a value of 1150            Goto Start // then go to jump mark “Start”</p>
Block	<p>This instruction allows the user to execute several instructions in one instruction. These</p>

Name	Description
	<p>assignments are restricted to the "Control word" and "Setpoints" objects. Depending on the configuration of the device and the purpose of use, the user can choose between "Control value with 1 setpoint", "Control value with 2 setpoints" or "Control value with 3 setpoints".</p> <p><b>Example:</b>            Block // Transmit control word and setpoint1 with 1 USS protocol            Device 00 Controlword = 1151 // Assign value of 1150 to control word            Device 00 Setpoint1 = 20.0 // Assign value of 20 to setpoint 1</p>
Mathematical and logical operations	<p>These instructions make it possible for the user to carry out simple mathematical and logical operations on objects. The newly-calculated value is then assigned to an object. The parameters of the instruction can be adapted in the "Properties" view.</p> <p><b>Example:</b>            Var1 = Device 00 Control word + 047F hex // Addition            Var1 = Device 00 Status word AND 047F hex // "And" operation</p>

### 8.3 Scheduler

#### Auto

With this option activated (automatic mode) after starting the scheduler line for line processing occurs. If it is deactivated (single step mode) (menu entry "Next" or combination of keys "F12 ") the user must run each instruction manually.

#### Loop

With this option activated the macro is implemented in a continuous loop. This means that after doing the last instruction the scheduler jumps back to the beginning of the macro.

#### 8.3.1 Start sequence

The scheduler is started using the "Start" menu item or key combination "F9". If automatic mode is active, processing takes place line by line. In single-step mode, only the first line is executed after starting. The user must call up the "Next" action for each subsequent line. The scheduler can only be started again after the macro has been worked through or the user has aborted the run. The parameters of the instructions cannot be edited whilst the scheduler is running.

#### 8.3.2 Cancel a macro

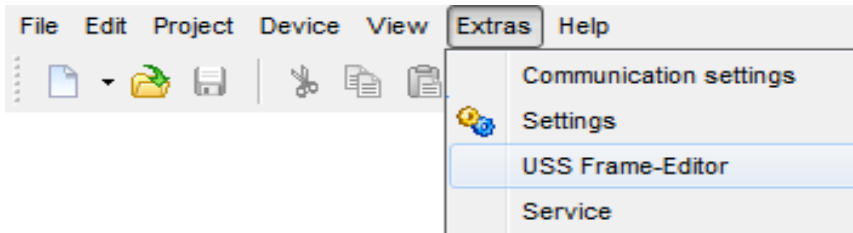
The scheduler is terminated in the menu option "Cancel" or the combination of keys "F11 ".

#### 8.3.3 Execute next instruction

This function can be found in the menu option "Next" or with the key "F12 ". It is available only in the single step mode and instructs the scheduler to implement the next instruction in the macro. If the last instruction was implemented, the scheduler is terminated automatically.

## 9 USS Frame-Editor

The USS protocol defines an access procedure according to the Master/Slave principle for communication via a serial bus. A sub-set of this also includes point-to-point connection. A master and a maximum of 31 slaves can be connected to a bus. The individual slaves are accessed by the master via an address character in the telegram. Direct exchange of messages between the individual slaves is not possible. In semi-duplex mode communication is carried out using USS telegrams.



The USS Frame Editor was developed to generate and analyse USS telegrams. It is fully integrated in the NORDCON user interface and is opened via the menu item "Extras/USS Frame-Editor". The editor displays the master and slave telegram in several views. Via the tabs, the user can switch between the and the 9.1 "Master (order)"Slave- Telegramm (USS Antwort) </dg\_ref\_source\_inline>.

Object	Description																					
Telegram type	<p>This object specifies the size and structure of the USS telegram. The frequency inverter supports the following types:</p> <table border="1"> <thead> <tr> <th>Type</th> <th>Length (LGE)</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>PPO 0</td> <td>12</td> <td>Standard telegram with process data and 16 bit parameter value</td> </tr> <tr> <td>PPO 1</td> <td>14</td> <td>Extended parameter data telegram with 32 bit parameter values and process data</td> </tr> <tr> <td>PPO 2</td> <td>18</td> <td>Telegram with extended process data (main and two auxiliary setpoint values) and 32 bit parameter value</td> </tr> <tr> <td>PPO 3</td> <td>6</td> <td>Process data telegram with main setpoint value without parameter data</td> </tr> <tr> <td>PPO 4</td> <td>10</td> <td>Extended process data telegram with main and auxiliary setpoint values without parameter data</td> </tr> <tr> <td>PPO 6</td> <td>16</td> <td>Telegram with 5 setpoint/actual values  <b>Notice:</b> This telegram type is not supported by all frequency inverters.</td> </tr> </tbody> </table>	Type	Length (LGE)	Description	PPO 0	12	Standard telegram with process data and 16 bit parameter value	PPO 1	14	Extended parameter data telegram with 32 bit parameter values and process data	PPO 2	18	Telegram with extended process data (main and two auxiliary setpoint values) and 32 bit parameter value	PPO 3	6	Process data telegram with main setpoint value without parameter data	PPO 4	10	Extended process data telegram with main and auxiliary setpoint values without parameter data	PPO 6	16	Telegram with 5 setpoint/actual values  <b>Notice:</b> This telegram type is not supported by all frequency inverters.
Type	Length (LGE)	Description																				
PPO 0	12	Standard telegram with process data and 16 bit parameter value																				
PPO 1	14	Extended parameter data telegram with 32 bit parameter values and process data																				
PPO 2	18	Telegram with extended process data (main and two auxiliary setpoint values) and 32 bit parameter value																				
PPO 3	6	Process data telegram with main setpoint value without parameter data																				
PPO 4	10	Extended process data telegram with main and auxiliary setpoint values without parameter data																				
PPO 6	16	Telegram with 5 setpoint/actual values  <b>Notice:</b> This telegram type is not supported by all frequency inverters.																				
Address	This object contains the address of the frequency inverter which is accessed.																					

9.2 "Device (response)"Zustandswort</dg_ref_source_inline>	This object contains the status bit of the frequency inverter.
5.3.5 "Status word"Steuerwort</dg_ref_source_inline>	This object contains the control bits (e.g. enable or Quick Stop).
Setpoint/actual value 1-5	The setpoint/actual values are 16 bit or 32 bit values. These represent different values (e.g. frequency setpoint or position setpoint) depending on the parameterisation of the frequency inverter.
Format	<p>This object contains the format of the setpoint. The following formats are supported:</p> <ul style="list-style-type: none"> <li>• 16 Bit standard value      This formatting interprets the setpoint as a 16 bit standardised value. Standardisation means scaling of the value range and is between -200% and 199% of a basic value (e.g. nominal frequency).</li> <li>• 16 Bit non-standardised      In this format the setpoint is interpreted as a 16 bit value, which is transferred to the frequency inverter and displayed without scaling.</li> </ul>
Parameter order	<p>The object contains the parameter order. The following orders are defined:</p> <ul style="list-style-type: none"> <li>• Request parameter value</li> <li>• Change parameter value (16 bit)</li> <li>• Change parameter value (32 bit)</li> <li>• Request parameter value (array)</li> <li>• Change parameter value (array 16 bit)</li> <li>• Change parameter value (array 32 bit)</li> <li>• Request the number of array elements</li> <li>• Change parameter value (array double word) without writing to the EEPROM</li> <li>• Change parameter value (array word) without writing to the EEPROM</li> <li>• Change parameter value (double word) without writing to the EEPROM</li> <li>• Change parameter value (word) without writing to the EEPROM</li> </ul>
Parameter number	This object contains the parameter number.
Index	The object contains the parameter index.
Value	This object contains the parameter value. Depending on the telegram type, this is a 16 or 32 bit value. The display of the value still depends on the resolution of the value.
Resolution	This object contains the resolution of the parameter. If the resolution is changed, only the display of the parameter value changes. Please refer to the frequency inverter instructions for the resolution value.

### Process value sequence 1,3,2 for SK700, SK300, Vector CT and VT

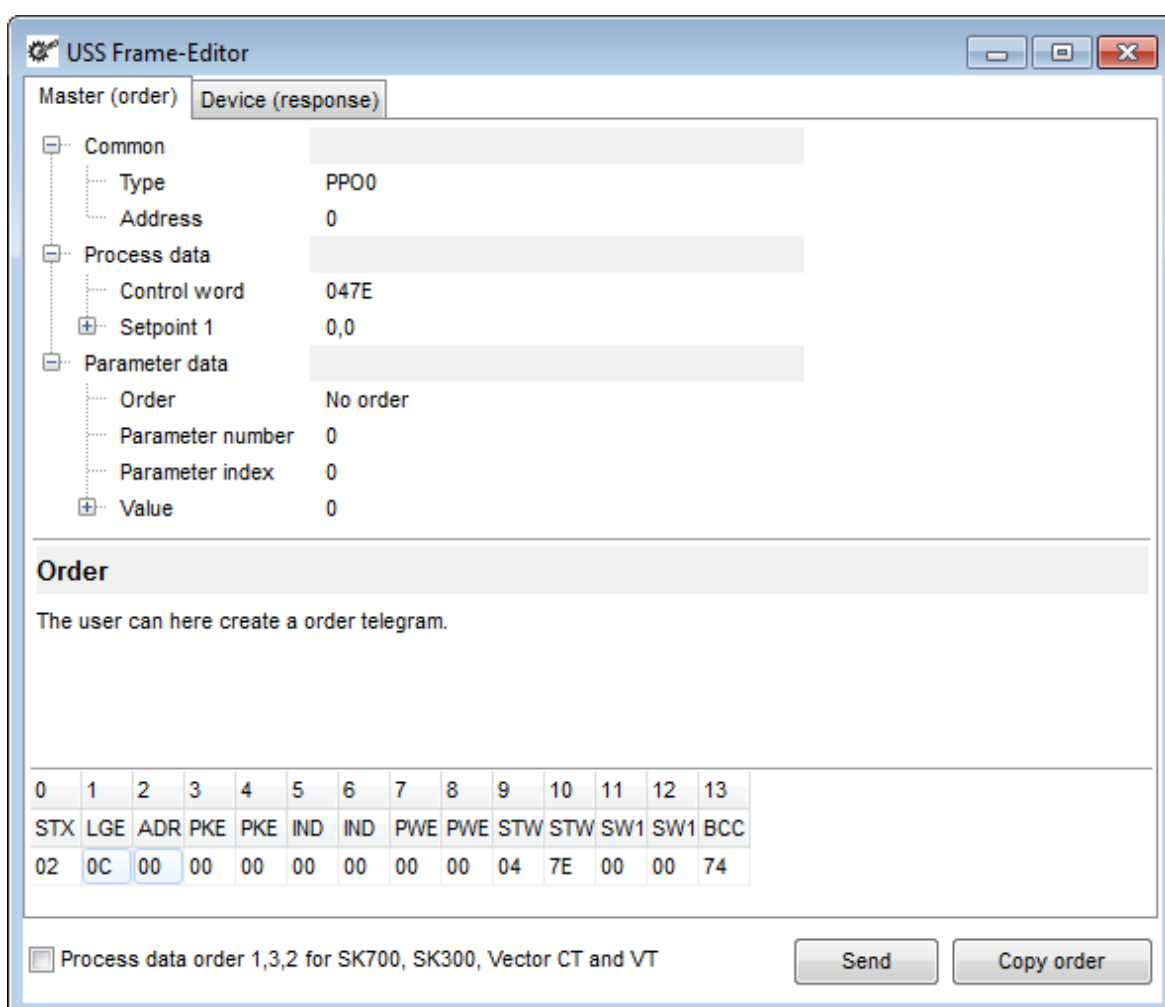
With this option, the sequence for the 2 and 3 process value can be changed for older frequency inverters. This option only affects the telegram types PPO 2 and PPO 4. The sequence of the process values is displayed in the table view.

### 5.3.6 "Control word"



## 9.1 Master (order)

This view is divided into several sections. In the upper section, the order telegram is displayed as a tree structure. The individual components of the telegram are listed by subject in the tree structure. All entries with a white background can be changed by the user. To do this, the entry must be highlighted with the mouse or the keyboard. A further click on the entry opens the input editor. The input editor may differ, depending on the entry. For numerical values, the input editor can also be opened by pressing a number key. The input for the new value is adopted and the input editor closed by pressing the "Enter" key or by highlighting a new entry. If the value cannot be adopted, the old value remains in use. If the input editor is a selection list, a new value is adopted by selecting an entry and the input editor is closed. If a change is not to be adopted, the user must exit from the input editor by pressing the "Esc" key. A description of each highlighted entry is displayed below the tree structure. In the lower section, the order telegram is displayed byte-wise in a table. The highlighted cells correspond to the entry which is highlighted in the tree structure.



The screenshot shows the 'USS Frame-Editor' window with the 'Master (order)' tab selected. The tree structure on the left lists the following components and their values:

- Common
  - Type: PPO0
  - Address: 0
- Process data
  - Control word: 047E
  - Setpoint 1: 0,0
- Parameter data
  - Order: No order
  - Parameter number: 0
  - Parameter index: 0
  - Value: 0

Below the tree structure, there is a section titled 'Order' with the text: 'The user can here create a order telegram.'

At the bottom, a table displays the telegram bytes:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
STX	LGE	ADR	PKE	PKE	IND	IND	PWE	PWE	STW	STW	SW1	SW1	BCC
02	0C	00	00	00	00	00	00	00	04	7E	00	00	74

At the bottom of the window, there is a checkbox labeled 'Process data order 1,3,2 for SK700, SK300, Vector CT and VT' and two buttons: 'Send' and 'Copy order'.

### Copy query

This action converts the order telegram into a hex coded byte string and copies the string to the Windows clipboard.

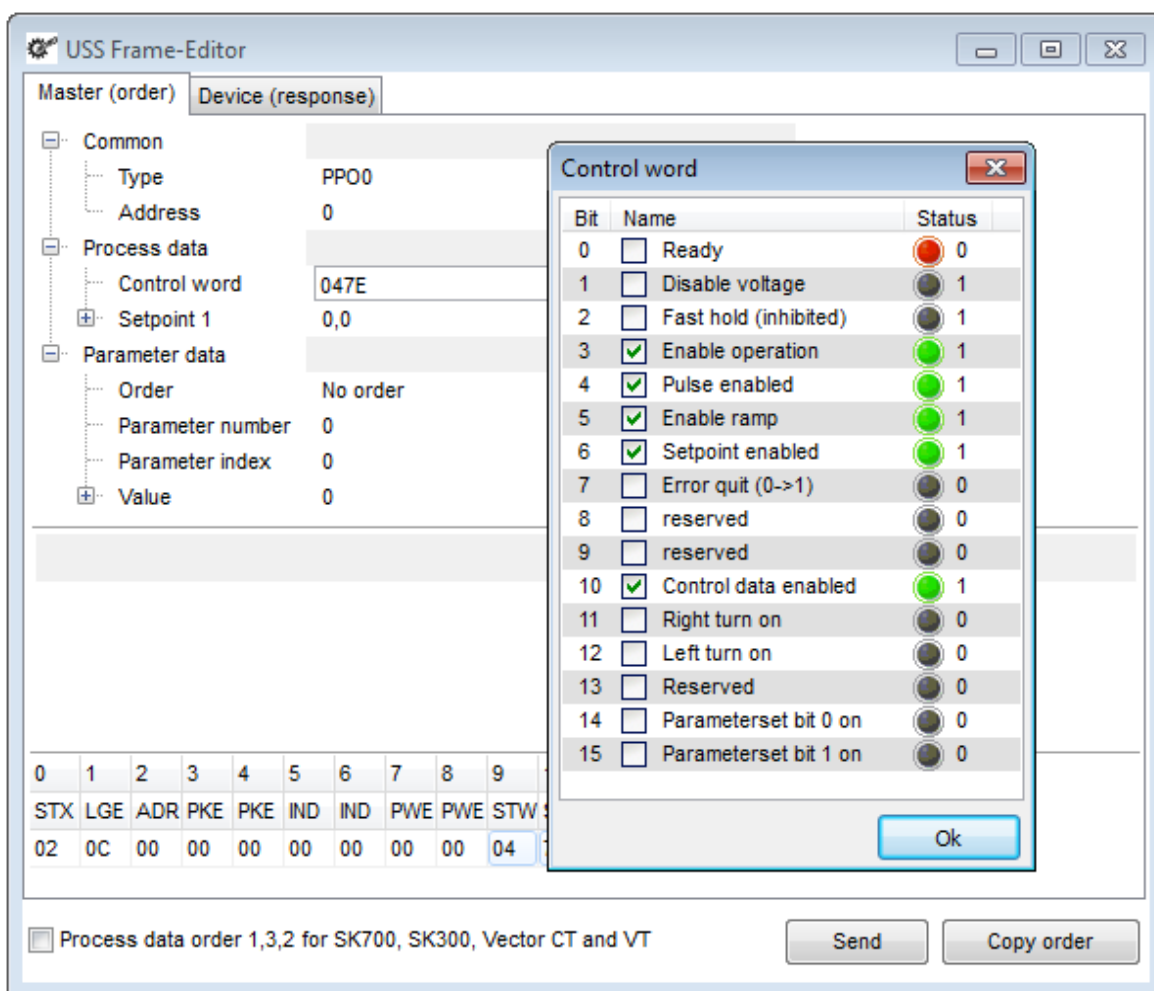
## 9.2 Device (response)

This view is divided into several sections. In the lower section, the response telegram is displayed byte-wise in a table. The user can change the response telegram in this table. All bytes except for

STX, LGE and BCC can be changed. The user selects a cell and enters a new value in the table. The context menu of the table must be opened if the length and structure of the telegram is to be changed. After this, a new telegram type is selected in the menu.

The tree structure is updated after each change. The tree structure is only used to visualise the components of the USS telegram and cannot be edited. An exception to this is the formatting of actual values and the resolution of the parameter value. This information is not contained in the USS telegram. The formatting must be changed according to the settings for the actual values. The resolution must also be selected according to the parameter. Please refer to the instructions for the particular frequency inverter to obtain the value.

The status word is displayed as a hexadecimal value in the tree structure. A further view is implemented for visualisation of the individual bits. The status word must be highlighted to open the view. A further click on the entry opens the input editor in write-protected mode. The user can then open the view with the "... " button.



### Enter response

This action opens an input dialogue for a response telegram. The user can enter the telegram as a hex-coded byte string.

## 10 PLC

### 10.1 General

The NORD frequency inverter series SK 180E/SK 190E, SK 2xxE, SK 2xxE-FDS and SK 5xxE as well as the motor starter series SK 155E-FDS/SK 175E-FDS contains logic processing which is similar to the current IEC61131-3 standard for memory programmable control units (SPS / PLC). The reaction speed or computing power of this PLC is suitable to undertake smaller tasks in the area of the inverter. Inverter inputs or information from a connected field bus can be monitored, evaluated and further processed into appropriate setpoint values for the frequency inverter. In combination with other NORD devices, visualisation of system statuses or the input of special customer parameters is also possible. Therefore, within a limited range, there is a potential for savings via the elimination of a previous external PC solution. AWL is supported as the programming language. AWL is a machine-orientated, text-based programming language whose scope and application is specified in IEC61131-3.

#### Information

Programming and download into the devices are possible exclusively via the NORD software NORD CON.

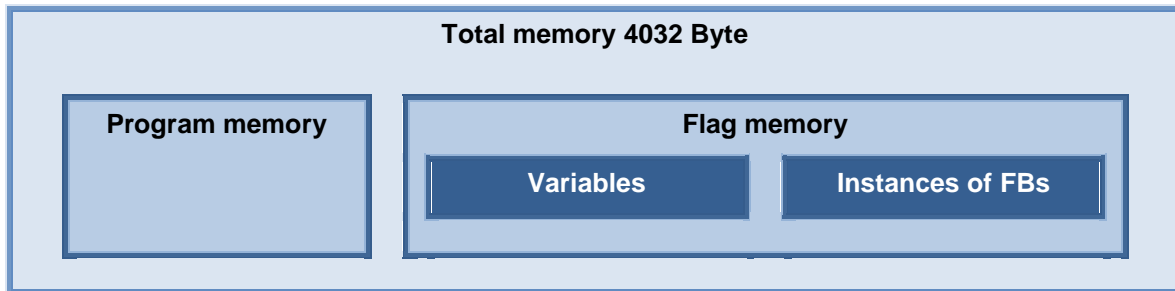
#### 10.1.1 Specification of the PLC

Function	Specification		
Standard	Orientated to IEC61131-3		
Language	Instruction List ( IL ), Structured Text (ST)		
Task	A cyclic task, program call-up every 5 ms		
Computer performance	Approximately 200 AWL commands per 1 ms		
Program memory	SK 5xxE, SK 2xxE, SK 2x0E-FDS	SK 190E / SK 180E	SK 155E FDS / SK 175E-FDS
	8128 Byte for flags, functions and the PLC program	2032 Byte for flags, functions and the PLC program	2028 Byte for flags, functions and the PLC program
Max. possible number of commands	Approximately 2580 commands	Approximately 660 commands	Approximately 660 commands
	<b>Note:</b> This is an average value. Heavy use of flags, process data and functions considerably reduces the possible number of lines; see Resources section.		
Freely accessible CAN mailboxes	20		
Supported devices	SK 54xE SK 53xE / SK 52xE ab V3.0 SK 2xxE ab V2.0 SK 2x0E-FDS SK 180E / SK 190E SK 155E-FDS / SK 175E-FDS		

### 10.1.2 PLC structure

#### 10.1.2.1 Memory

The PLC memory is divided into the program memory and the flag memory. In addition to the variables, instances of function blocks are saved in the area of the flag memory. FB instance is a memory area in which all internal input and output variables of function command are saved. Each function command declaration requires a separate instance. The boundary between the program memory and the flag memory is determined dynamically, depending on the size of the flag area.



In the flag memory, two different classes of variables are stored in the variable section:

#### [VAR]

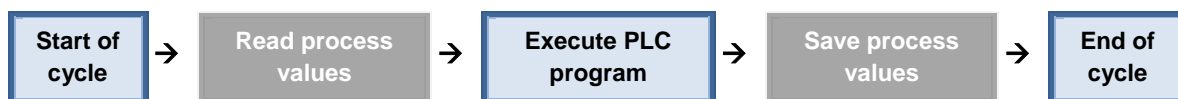
Memory variable for saving auxiliary information and statuses. Variables of this type are initialised every time the PLC starts. The memory content is retained during the cyclic sequence of the PLC.

#### [VAR\_ACCESS]

These are used to read and describe process data (inputs, outputs, setpoints, etc.) of the frequency inverter. These values are regenerated with every PLC cycle.

#### 10.1.2.2 Image of the process

Several physical dimensions such as torque, speed, position, inputs, outputs etc. are available to the device. These dimensions are divided into actual and setpoint values. They can be loaded into the process image of the PLC and influenced by it. The required processes must be defined in the list of variables under the class VAR\_ACCESS. With each PLC cycle, all of the process data for the inverter which is defined in the list of variables is newly read in. At the end of each PLC cycle the writable process data are transferred back to the inverter, see following illustration.



Because of this sequence it is important to program a cyclic program sequence. Programming loops in order to wait for a certain event (e.g. change of level at an input) does not produce the required result. This behaviour is different in the case of function blocks which access process values. Here, the process value is read on call-up of the function block and the process values are written immediately when the block is terminated.

### **i** Information

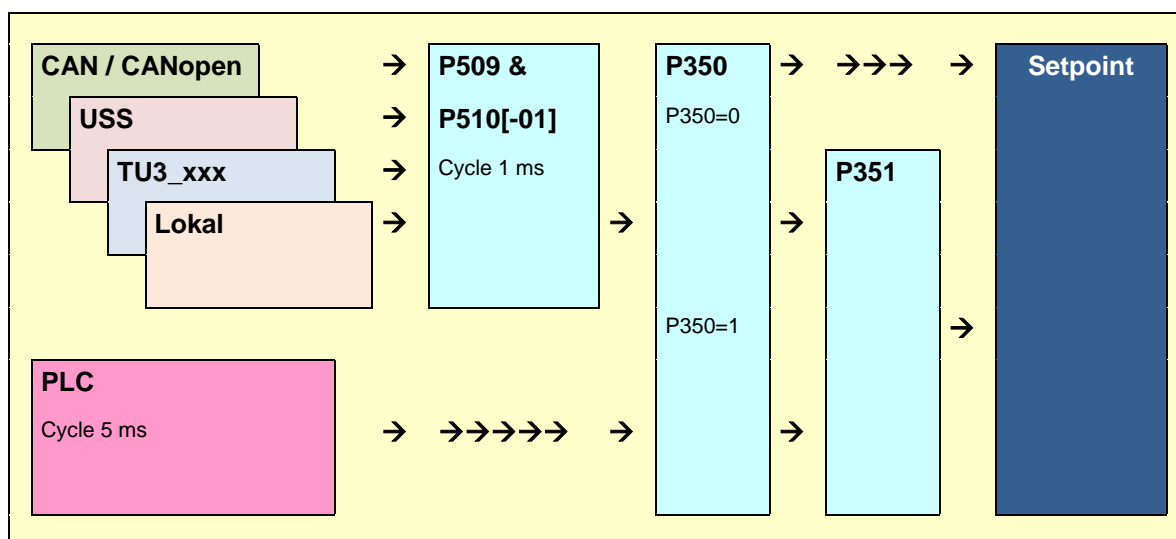
If the Motion blocks MC\_Power, MC\_Reset, MC\_MoveVelocity, MC\_Move, MC\_Home or MC\_Stop are used, the process values "PLC\_Control\_Word" and "PLC\_Set\_Val1" up to "PLC\_Set\_Val5" may not be used. Otherwise the values in the list of variables would always overwrite the changes to the function block..

#### 10.1.2.3 Program Task

Execution of the program in the PLC is carried out as a single task. The task is called up cyclically every 5 ms and its maximum duration is 3 ms. If a longer program cannot be executed in this time, the program is interrupted and continued in the next 5 ms task.

#### 10.1.2.4 Setpoint processing

The inverter has a variety of setpoint sources, which are ultimately linked via several parameters to form a frequency inverter setpoint.



If the PLC is activated (P350=1) preselection of setpoints from external sources (main setpoints) is carried out via P509 and P510[-01] Via P351, a final decision is made as to which setpoints from the PLC or values input via P509/P510[-01] are used. A mixture of both is also possible. No changes to the auxiliary setpoints (P510[-02]) are associated with the PLC function. All auxiliary setpoint sources and the PLC transfer their auxiliary setpoint to the frequency inverter with equal priority.

#### 10.1.2.5 Data processing via accumulator

The accumulator forms the central computing unit of the PLC. Almost all AWL commands only function in association with the accumulator. The PLC has three accumulators. These are the 23 Bit Accumulator 1 and Accumulator 2 and the AE in BOOL format. The AE is used for all boolean loading, saving and comparison operations. If a boolean value is loaded, it is depicted in the AE Comparison operations transfer their results to the AE and conditional jumps are triggered by the AE. Accumulator 1 and Accumulator 2 are used for all operands in the data format BYTE, INT and DINT. Accumulator 1 is the main accumulator and Accumulator 2 is only used for auxiliary functions. All loading and storage operands are handled by Accumulator 1. All arithmetic operands save their results in Accumulator 1. With each Load command, the contents of Accumulator 1 are moved to Accumulator 2. A subsequent operator can link the two accumulators together or evaluate them and save the result in Accumulator 1, which in the following will generally be referred to as the "accumulator".

### 10.1.3 Scope of functions

The PLC supports a wide range of operators, functions and standard function modules, which are defined in IEC61131-3. There is a detailed description in the following sections. In addition, the function blocks which are also supported are explained.

#### 10.1.3.1 Motion Control Lib

The Motion Control Lib is based on the PLCopen specification "Function blocks for motion control". This mainly contains function blocks which are used to move the drive. In addition, function blocks for reading and writing of parameters of the device are also provided.

#### 10.1.3.2 Electronic gear with Flying Saw

The frequency inverter is equipped with the functions Electronic gear unit (synchronous operation in positioning mode) and Flying saw. Via these functions the inverter can follow another drive unit with angular synchronism. As well as this, with the additional function Flying saw it is possible to synchronize to the precise position of a moving drive unit. The operating mode Electronic gear unit can be started and stopped at any time. This enables a combination of conventional position control with its move commands and gear unit functions. For the gear function a NORDAC vector with internal CAN bus is required on the master axis.

#### 10.1.3.3 Visualisation

Visualisation of the operating status and the parameterisation of the frequency inverter is possible with the aid of a ControlBox or a ParameterBox. Alternatively, the CANopen Master functionality of the PLC CAN bus panel can be used to display information.

##### **ControlBox**

The simplest version for visualisation is the ControlBox. The 4-digit display and the keyboard status can be accessed via two process values. This enables simple HMI applications to be implemented very quickly. P001 must be set to "PLC-ControlBox Value" so that the PLC can access the display. A further special feature is that the parameter menu is no longer accessed via the arrow keys. Instead, the "On" and "Enter" keys must be pressed simultaneously.

##### **ParameterBox**

In visualisation mode, each of the 80 characters in the ParameterBox display (4 rows of 20 characters) can be set via the PLC. It is possible to transfer both numbers and texts. In addition keyboard entries on the ParameterBox can be processed by the PLC. This enables the implementation of more complex HMI functions (display of actual values, change of window, transfer or setpoints etc.). Access to the ParameterBox display is obtained via the function blocks in the PLC. Visualisation is via the operating value display of the Parameter Box. The content of the operating value display is set via the ParameterBox parameter P1003. This parameter can be found under the main menu item "Display". P1003 must be set to the value "PLC display". After this, the operating value display can be selected again by means of the right and left arrow keys. The display controlled by the PLC is then shown. This setting remains in effect even after a further switch-on.

#### 10.1.3.4 Process controller

The process controller is a PID-T1 controller with a limited output size. With the aid of this function module in the PLC it is possible to simply set up complex control functions, by means of which various processes, e.g. pressure regulation, can be implemented in a considerably more elegant manner than with the commonly used two-point controllers.

### 10.1.3.5 CANopen communication

In addition to the standard communication channels, the PLC provides further possibilities for communication. Via the CAN bus interface of the frequency inverter, it can set up additional communications with other devices. The protocol which is used for this is CANopen. Communications are restricted to PDO data transfer and NMT commands. The standard CANopen inverter communication via SDO, PDO1, PDO2 and Broadcast remains unaffected by this PLC function.


#### **PDO (Process Data Objects)**

Other frequency inverters can be controlled and monitored via PDO. However, it is also possible to connect devices from other manufacturers to the PLC. These may be IO modules, CANopen encoders, panels, etc. With this, the number of inputs/outputs of the frequency encoder can be extended as far as is required; analog outputs would then be possible.

#### **NMT (Network Management Objects)**

All CANopen devices must be set to the CANopen bus state "Operational" by the bus master. PDO communication is only possible in this bus state. If there is no bus master in the CANopen bus, this must be performed by the PLC. The function module FB\_NMT is available for this purpose.

## 10.2 Creation of PLC programs

Creation of the PLC programs is carried out exclusively via the PC program NORD CON. The PLC editor is opened either via the menu item "File/New/PLC program" or via the symbol . This button is only active if a device with PLC functionality forms the focus of the device overview.

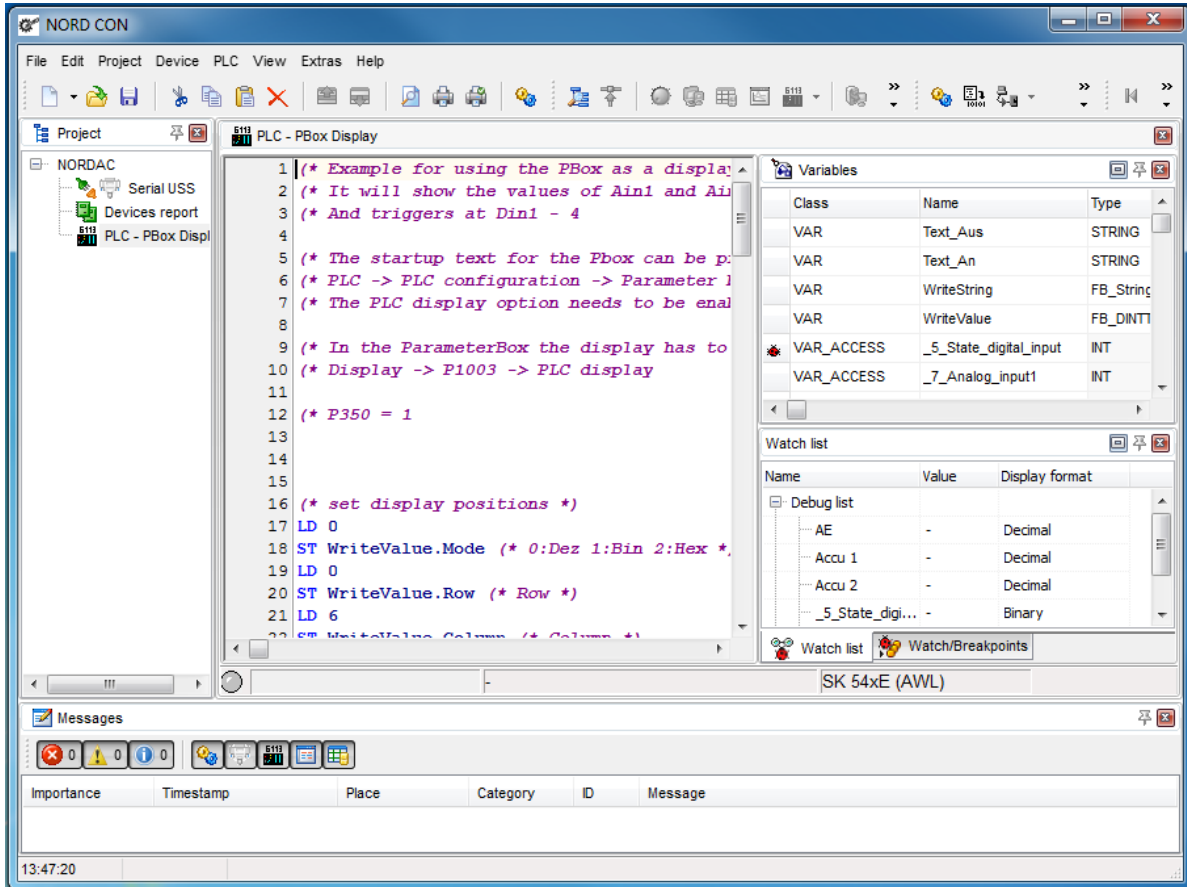
### 10.2.1 Loading, saving and printing

The functions Load, Save and Print are carried out via the appropriate entries in the main menu or in the symbol bars. When opening takes place, it is advisable to set the file type to "PLC Program" (\*.awlx) in the "Open" dialogue. With this, only files which can be read by the PLC editor are displayed. If the PLC program which has been created is to be saved, the PLC Editor window must be active. The PLC program is saved by actuating "Save" or "Save as". With the "Save as" operation, this can also be detected from the entry of the file type (Program PLC (\*.awlx)). The appropriate PLC window must be active in order to print the PLC program. Printout is then started via "File/Print" or the appropriate symbol.

PLC programs can also be saved as a backed-up PLC program. To do this, the user must set the file type to "Backed-up AWL files" or "Backed up ST files" in the file selection dialogue. Then the PLC program is saved in an encrypted (\*.awls or \*.nsts) and normal (\*.awlx, \*.nstx) version. The encrypted PLC program can only be transmitted to the device (see 2.2.4 "Category "Device"").

### 10.2.2 Editor

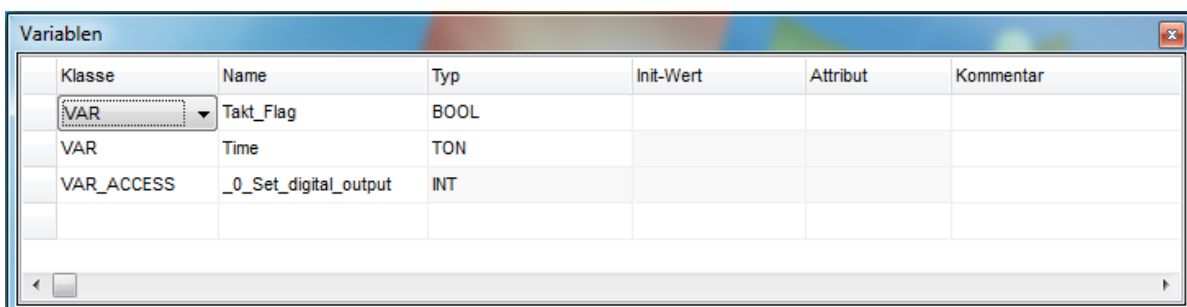
The PLC Editor is divided into four different windows.



The individual windows are described in more detail in the following sections.

### 10.2.2.1 Variables and FB declaration

All the variables, process values and function blocks which are required by the program are declared in this window.



#### Variables

Variables are created by setting the Class "VAR". The Name of the variable can be freely selected. In the Type field, a selection between BOOL, BYTE, INT and DINT can be made. A starting initialisation can be entered under Init-Value.

#### Process values

These are created by selecting the entry "VAR\_ACCESS" under Class. The Name is not freely selectable and the field Init-Value is barred for this type.



## Function modules

The entry "VAR" is selected under Class. The Name for the relevant instance of the function module (FB) can be freely selected. The required FB is selected under Type. An Init-Value cannot be set for function modules.

All menu items which relate to the variable window can be called up via the context menu. Via this, entries can be added and deleted. Variables and process variables for monitoring (Watchdog function) or debugging (Breakpoint) can be activated.

### 10.2.2.2 Input window

The input window is used to enter the program and to display the AWL program. It is provided with the following functions:

- Highlight syntax
- Bookmark
- Declaration of variables
- Debugging

### Syntax Highlighting

If the command and the variable which is assigned to it are recognised by the Editor, the command is displayed in blue and the variable in black. As long as this is not the case, the display is in thin black italics.

### Bookmarks

As programs in the Editor may be of considerable length, it is possible to mark important points in the program with the function Bookmark and to jump directly to these points. The cursor must be located in the relevant line in order to mark it. Via the menu item "Switch bookmark" (right mouse button menu) the line is marked with the required bookmark. The bookmark is accessed via the menu item "Go to bookmark".

### Declaring Variables

Via the Editor menu "Add Variable" (right mouse button) new variables can be declared using the Editor.

### Debugging

For the Debugging function, the positions of the breakpoints and watchpoints are specified in the Editor. This can be done via the menu items "Switch breakpoint" (Breakpoints) and "Switch monitoring point" (Watchpoints). The position of Breakpoints can also be specified by clicking on the left border of the Editor window. Variables and process values which are to be read out from the frequency inverter during debugging must be marked. This can be done in the Editor via the menu items "Debug variable" and "Watch variable". For this, the relevant variable must be marked before the required menu item is selected.

### 10.2.2.3 Watch and Breakpoint display window

This window has two tabs, which are explained below.

### Holding points

This window displays all of the breakpoints and watchpoints which have been set. These can be switched on and off via the checkboxes and deleted with the "Delete key". A corresponding menu can be called up with the right mouse button.

### **Observation list**

This displays all of the variables which have been selected for observation. The current content is displayed in the Value column. The display format can be selected with the Display column.

#### **10.2.2.4 PLC message window**


All PLC status and error messages are entered in this window. In case of a correctly translated program the message "Translated without error" is displayed. The use of resources is shown on the line below this. In case of errors in the PLC program, the message "Error X" is displayed. The number of errors is shown in X. The following lines show the specific error message in the format:

[Line number]: Error description


#### **10.2.3 Transfer PLC program to device**

There are several ways to transfer a PLC program to the device.


##### **Transfer PLC program directly:**

1. Select device in the project tree
2. Open popup menu (press the right mouse button)
3. Execute function "Transfer PLC program to device" 
4. Select file in the file selection dialogue and press "Open"

##### **Transfer PLC program with PLC editor (offline):**

1. Open PLC program (File->Open)
2. Connect PLC editor with a device (PLC->Connect)
3. Translate PLC program
4. Transfer PLC program to device 

##### **Transfer PLC program with PLC editor (online):**

1. Select device in the project tree
2. Open PLC editor 
3. Open PLC program

4. Import the file into online view
5. Translate PLC program
6. Transfer PLC program to device

---

## Information

### SK 1xxE-FDS – limited number of writing cycles

In the devices SK 155E-FDS / SK 175E-FDS flash is used as a storage medium. The number of write cycles of Flash memory is very limited. By default, the program is loaded into the RAM. It can then be started and tested. If the PLC is then restarted, the program must be re-loaded to the device to initialize the PLC variables. Should the program be permanently stored in the device, you must execute the function "Transfer and store program to device".

---

## 10.2.4 Debugging

As programs only rarely function the very first time, the PLC provides several possibilities for finding faults. These possibilities can be roughly divided into two categories, which are described in detail below.

### 10.2.4.1 Observation points (Watchpoints)

The simplest debugging variant is the Watchpoint function. This provides a rapid overview of the behaviour of several variables. For this, an observation point is set at an arbitrary point in the program. When the PLC processes this line, up to 5 values are saved and displayed in the observation list (window "Observation List") The 5 values to be observed can be selected in the entry window or in the variable window using the context menu.

---



## Information

In the current version, variables of functions cannot be added to the watch list!

---



### 10.2.4.2 Holding points (Breakpoints)

Via holding points it is possible to deliberately stop the PLC program at a specific line of the program. If the PLC runs into a Breakpoint, the AE, Accumulator 1 and Accumulator 2 are read out, as well as all variables which have been selected via the menu item "Debug variables". Up to 5 Breakpoints can



be set in a PLC  program. This function is started via the  symbol. The program now runs until a holding point is triggered. Further actuation of the symbol bar allows the program to continue running until it reaches the next holding point. If the program is to continue running, the symbol is actuated.

### 10.2.4.3 Single Step

With this debugging method it is possible to execute the PLC program line for line. With each individual step, all the selected variables are read out of the PLC of the device and displayed in the "Observation list" window. The values to be observed can be selected in the input window or the variable window by means of the right mouse button menu. The condition for debugging in single steps is that at least one Breakpoint has been set before starting debugging. The debugging mode is

switched on by actuating the  symbol. Only when the program has run into the first breakpoint, can the following lines be debugged via the  symbol. Some command lines contain several

individual commands. Because of this, two or more individual steps may be processed before the step indicator jumps forward in the entry window. The actual position is shown by a small arrow in the left

PLC Editor window. When the  symbol is actuated, the program continues running until the next holding point. If the program is to continue running, the  symbol is actuated.

### 10.2.5 PLC configuration

The PLC configuration dialogue is opened via the  symbol. Here, basic settings for the PLC can be made, which are described in further detail below.

#### Cycle time monitoring

This function monitors the maximum processing time for a PLC cycle. With this, unintended continuous program loops in the PLC program can be caught. Error E22.4 is triggered in the frequency inverter if this time is exceeded.

#### Allow ParameterBox function module

If visualisation via the ParameterBox is to be performed in the PLC program, this option must be enabled. Otherwise the corresponding function blocks generate a Compiler Error when the frequency inverter is started.

#### Invalid control data

The PLC can evaluate control words which are received from the possible bus systems. However, the control words can only get through if the bit "PZD valid" (Bit 10) is set. This option must be activated if control words which are not compliant with the USS protocol are to be evaluated by the PLC. Bit 10 in the first word is then no longer queried.

#### Do not pause the system time at holding point

The system time is paused during debugging if the PLC is in the holding point or in single step mode. The system time forms the basis for all timers in the PLC. This function must be activated if the system time is to continue running during debugging.

## 10.3 Function blocks

Function blocks are small programs, which can save their status values in internal variables. Because of this, a separate instance must be created in the NORD CON variable list for each function block. E.g. if a timer is to monitor 3 times in parallel, it must also be set up three times in the list of variables.

---

### Information

#### Detecting a signal edge

In order for the following function blocks to detect an edge at the input, it is necessary for the function call-up to be carried out twice with different statuses at the input.

---

#### 10.3.1 CANopen

The PLC can configure, monitor and transmit on PDO channels via function blocks. The PDO can transmit or receive up to 8 bytes of process data via a PDO. Each of these PDOs is accessed via an individual address (COB-ID). Up to 20 PDOs can be configured in the PLC. For simpler operation, the COB-ID is not entered directly. Instead, the device address and the PDO number are communicated

to the FB. The resulting COB-ID is determined on the basis of the Pre-Defined Connection Set (CiA DS301). This results in the following possible COB-IDs for the PLC.

Transmit PDO		Receive PDO	
PDO	COB-ID	PDO	COB-ID
PDO1	200h + Device address	PDO1	180h + Device address
PDO2	300h + Device address	PDO2	280h + Device address
PDO3	400h + Device address	PDO3	380h + Device address
PDO4	500h + Device address	PDO4	480h + Device address

NORD Frequency inverter use PDO1 to communicate process data. PDO2 is only used for setpoint/actual value 4 and 5.

### 10.3.1.1 Overview

Function module	Description
FB_PDConfig	PDO configuration
FB_PDOSend	Transmit PDO
FB_PDORceive	Receive PDO
FB_NMT	Enable and disable PDO

### 10.3.1.2 FB\_NMT

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

After a Power UP all CAN participants are in the bus state Pre-Operational. In this state, they can neither transmit nor receive a PDO. In order for the PLC to be able to communicate with other participants on the CAN bus, these must be set to an operational state. Usually, this is performed by the bus master. If there is no bus master, this task can be performed by the FB\_NMT. The status of all of the participants connected to the bus can be controlled via the inputs **PRE**, **OPE** or **STOP**. The inputs are adopted with a positive flank on **EXECUTE**. The function must be called up until the output **DONE** or **ERROR** has been set to 1.

If the **ERROR** is set to 1, there is either no 24V supply to the RJ45 CAN socket of the inverter, or the CAN driver of the inverter is in the status Bus off. With a negative flank on **EXECUTE**, all outputs are reset to 0.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Execute	BOOL	<b>DONE</b>	NMT command is transmitted	BOOL

<b>PRE</b>	Sets all participants to Pre-Operational status	BOOL	<b>ERROR</b>	Error in FB	BOOL
<b>OPE</b>	Sets all participants to Operational status	BOOL			
<b>STOP</b>	Sets all participants to Stopped status	BOOL			

### 10.3.1.3 FB\_PDOConfig

	<b>SK 54xE</b>	<b>SK 53xE SK 52xE</b>	<b>SK 2xxE</b>	<b>SK 2xxE-FDS</b>	<b>SK 180E SK 190E</b>	<b>SK 155E-FDS SK 175E-FDS</b>
<b>Availability</b>	X	X	X	X		

The PDOs are configured with this FB. With an instance of this function, all of the required PDOs can be configured. The FB must only be called up once for each PDO. Up to 20 PDOs can be set up. Each PDO has its own parameterisation. Assignment of the PDOs in the other CANopen FBs is carried out via the message box Number. The **TARGETID** represents the address of the device. With NORD frequency inverters, this is set in P55 or via DIP switches. The required message box number is entered under PDO (see Introduction). **LENGTH** specifies the transmission length of a PDO. The transmission/reception direction is specified with **DIR**. The data are adopted with a positive flank on the **EXECUTE** input. The **DONE** output can be queried immediately after the call-up of the FB. If **DONE** is set to 1, the PDO channel has been configured. If **ERROR** = 1 there has been a problem, whose precise cause is saved in **ERRORID**. With a negative flank on **EXECUTE**, all outputs are reset to 0.

<b>Transmit PDO</b>		<b>Receive PDO</b>	
<b>PDO</b>	<b>COB-ID</b>	<b>PDO</b>	<b>COB-ID</b>
PDO1	200h + device address	PDO1	180h + device address
PDO2	300h + device address	PDO2	280h + device address
PDO3	400h + device address	PDO3	380h + device address
PDO4	500h + device address	PDO4	480h + device address
PDO5	180h + device address	PDO5	200h + device address
PDO6	280h + device address	PDO6	300h + device address
PDO7	380h + device address	PDO7	400h + device address
PDO8	480h + device address	PDO8	500h + device address

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Execute	BOOL	<b>DONE</b>	PDO configured	BOOL
<b>NUMBER</b>	Message box number Value range = 0 to 19	BYTE	<b>ERROR</b>	Error in FB	BOOL
<b>TARGETID</b>	Device address Value range = 1 to 127	BYTE	<b>ERRORID</b>	Error code	INT
<b>PDO</b>	PDO Value range = 1 to 4	BYTE			
<b>LENGTH</b>	PDO length Value range = 1 to 8	BYTE			
<b>DIR</b>	Transmit or receive Transmit = 1 / Receive = 0	BOOL			
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1800h	Number value range exceeded				
1801h	TARGETID value range exceeded				
1802h	PDO value range exceeded				
1803h	LENGTH value range exceeded				

### Information

### No duplicate use of CAN ID

CAN-IDs which are already being used by the device may not be parameterised!

Relevant reception addresses:

- CAN ID = 0x180 + P515[-01]                      PDO1
- CAN ID = 0x180 + P515[-01]+1                CAN ID for absolute encoders
- CAN ID = 0x280 + P515[-01]                      PDO2

Relevant transmission addresses:

- CAN ID = 0x200 + P515[-01]                      PDO1
- CAN ID = 0x300 + P515[-01]                      PDO2

**Example in ST:**

```

(* Configure PDO *)
PDOConfig(
  Execute := TRUE,
  (* Configure Message box 1 *)
  Number := 1,
  (* Set CAN node number *)
  TargetID := 50,
  (* Select (Standard for PDO1 control word, setpoint1, setpoint2, setpoint3) *)
  PDO := 1,
  (* Specify length of data (Standard for PDO1 is 8 *)
  LENGTH := 8,
  (* Transmit *)
  Dir := 1);

(* Configure PDO *)
PDOConfig(
  Execute := TRUE,
  (* Configure message box 1 *)
  Number := 2,
  (* Set CAN node number *)
  TargetID := 50,
  (* Select PDO (Standard for PDO2 setpoint4, setpoint5 SK540E) *)
  PDO := 2,
  (* Specify length of data (Standard for PDO2 is 4 *)
  LENGTH := 4,
  (* Transmit *)
  Dir := 1);

(* Configure PDO *)
PDOConfig(
  Execute := TRUE,
  (* Configure message box 2 *)
  Number := 2,
  (* Set CAN node number *)
  TargetID := 50,
  (* Select PDO (Standard for PDO1 status word, actual value1, actual value2, actual
  value3) *)
  PDO := 1,
  (* Specify length of data (Standard for PDO1 is 8 *)
  LENGTH := 8,
  (* Receive *)
  Dir := 0);

```

**10.3.1.4 FB\_PDOReceive**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X		

This FB monitors a previously configured PDO channel for incoming messages. Monitoring starts if the “**ENABLE**” input is set to 1. After the function has been called up, the **NEW** output must be checked. If it changes to 1, a new message has arrived. The **NEW** output is deleted with the next call-up of the function. The data which have been received are in **WORD1** to **WORD4**. The PDO channel can be monitored for cyclical reception via **TIME**. If a value between 1 and 32767 ms is entered in **TIME**, a message must be received within this period. Otherwise the FB goes into the error state (**ERROR** = 1). This function can be disabled with the value 0. The monitoring timer runs in 5 ms steps. In case of error **ERROR** is set to 1. **DONE** is 0 in this case. In the **ERRORID** the relevant error code is then valid. With a negative flank on **ENABLE**, **DONE**, **ERROR** and **ERRORID** are reset.



VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Execute	BOOL	<b>NEW</b>	PDO transmitted = 1	BOOL
<b>NUMBER</b>	Messagebox number Value range = 0 to 19	BYTE	<b>ERROR</b>	Error in FB	BOOL
<b>TIME</b>	Watchdog function Value range = 0 to 32767 0 = Disabled 1 to 32767 = Monitoring time t	INT	<b>ERRORID</b>	Error code	INT
			<b>WORD1</b>	Received data Word 1	INT
			<b>WORD2</b>	Received data Word 2	INT
			<b>WORD3</b>	Received data Word 3	INT
			<b>WORD4</b>	Received data Word 4	INT
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1800h	Number value range exceeded				
1804h	Selected box is not configured correctly				
1805h	No 24V for bus driver or bus driver is in "Bus off" status				
1807h	Reception timeout (Watchdog function)				

**i Information**

**PLC cycle time**

The PLC cycle is about 5 ms, i.e. with one call-up of the function in the PLC program, a CAN message can only be read every 5 ms. Messages may be overwritten if several messages are sent in quick succession.

**Example in ST:**

```

IF bFirstTime THEN
  (* Set the device to the status Pre-Operational *)
  NMT(Execute := TRUE, OPE := TRUE);
  IF not NMT.Done THEN
    RETURN;
  END_IF;

  (* Configure PDO *)
  PDOConfig(
    Execute := TRUE,
    (* Configure Messagebox 2 *)
    Number := 2,
    (* Set CAN node number *)
    TargetID := 50,
    (* Select PDO (Standard for PDO1 status word, actual value1, actual value2, actual
value3) *)
    PDO := 1,
    (* Specify length of data (Standard for PDO1 is 8 *)
    Length := 8,
    (* Receive *)
    Dir := 0);
  END_IF;

  (* Read out status and actual values *)
  PDOReceive(Enable := TRUE, Number := 2);
  IF PDOReceive.New THEN

```

```

State := PDOReceive.Word1;
Sollwert1 := PDOReceive.Word2;
Sollwert2 := PDOReceive.Word3;
Sollwert3 := PDOReceive.Word4;
END_IF

```

10.3.1.5 FB\_PDOSend

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X		

With this FB, PDOs can be transmitted on a previously configured channel. This is possible either as a one-off or cyclical transmission. The data to be transmitted is entered in **WORD1** to **WORD4**. Transmission of the PDO is possible regardless of the CANopen state of the frequency inverter. The previously configured PDO channel is selected via **NUMBER**. The data to be transmitted is entered into **WORD1** to **WORD4**. Via **CYCLE** one-off transmission (setting = 0) or cyclical transmission can be selected. The PDO is sent with a positive flank on **EXECUTE**. IF **DONE** = 1 all entries were correct and the PDO is sent. IF **ERROR** = 1 there was a problem. The precise cause is saved in **ERRORID**. All outputs are reset with a negative flank on **EXECUTE**. The time base of the PLC is 5ms; this also applies for the **CYCLE** input. Only transmission cycles with a multiple of 5ms can be implemented.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Execute	<b>BOOL</b>	<b>DONE</b>	PDO transmitted = 1	<b>BOOL</b>
<b>NUMBER</b>	Messagebox number Value range = 0 to 19	<b>BYTE</b>	<b>ERROR</b>	Error in FB	<b>BOOL</b>
<b>CYCLE</b>	Transmission cycle Value range = 0 to 255 0 = Disabled 1 to 255 = Transmission cycle in ms	<b>BYTE</b>	<b>ERRORID</b>	Error code	<b>INT</b>
<b>WORD1</b>	Transmission data Word 1	<b>INT</b>			
<b>WORD2</b>	Transmission data Word 2	<b>INT</b>			
<b>WORD3</b>	Transmission data Word 3	<b>INT</b>			
<b>WORD4</b>	Transmission data Word 4	<b>INT</b>			
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1800h	Number value range exceeded				
1804h	Selected box is not configured correctly				
1805h	No 24V for bus driver or bus driver is in "Bus off" status				

If **DONE** changes to 1, the message to be transmitted has been accepted by the CAN module, but has not yet been transmitted. The actual transmission runs in parallel in the background. If several messages are now to be sent directly in sequence via an FB, it may be the case that on the new call-up the previous message has not yet been sent. This can be identified by the fact that neither **DONE** nor the **ERROR** signal have been set to 1 after the CAL call-up. The CAL call-up can be repeated until one of the two signals changes to 1. If several different CAN IDs are to be written via a single FB, this is possible with a new configuration of the FB. However, this must not be done in the same PLC cycle as the transmission. Otherwise there is a danger that the message which is to be transmitted will be deleted by the FB\_PDOConfig.

### Example in ST:

```

IF bFirstTime THEN
  (* Set the device to the status Pre-Operational *)
  NMT(Execute := TRUE, OPE := TRUE);
  IF not NMT.Done THEN
    RETURN;
  END_IF;

  (* Configure PDO*)
  PDOConfig(
    Execute := TRUE,
    (* Configure Messagebox 1 *)
    Number := 1,
    (* Set CAN node number *)
    TargetID := 50,
    (* Select PDO (Standard for PDO1 control word, setpoint1, setpoint2, setpoint3) *)
    PDO := 1,
    (* Specify length of data (Standard for PDO1 is 8 *)
    LENGTH := 8,
    (* Transmit *)
    Dir := 1);

  IF not PDOConfig.Done THEN
    RETURN;
  END_IF;

  (* Transmit PDO - Set device to status Ready for Operation *)
  PDOSend(Execute := TRUE, Number := 1, Word1 := 1150, Word2 := 0, Word3 := 0, Word4 := 0);
  IF NOT PDOSend.Done THEN
    RETURN;
  END_IF;

  PDOSend(Execute := FALSE);
  bFirstTime := FALSE;
END_IF;

CASE State OF
  0:
    (* If digital input 1 is set *)
    IF _5_State_digital_input.0 THEN
      (* Transmit PDO - Set device to status Ready for Operation *)
      PDOSend(Execute := TRUE, Number := 1, Word1 := 1150, Word2 := 0, Word3 := 0,
        Word4 := 0);
      State := 10;
      RETURN;
    END_IF;

    (* If digital input 2 is set *)
    IF _5_State_digital_input.1 THEN
      (* Transmit PDO - Enable device and setpoint frequency to 50% of the maximum
        frequency *)
      PDOSend(Execute := TRUE, Number := 1, Word1 := 1151, Word2 := 16#2000, Word3 := 0,
        Word4 := 0);
      State := 10;
      RETURN;
    END_IF;

  10:
    PDOSend;
    IF PDOSend.Done THEN
      PDOSend(Execute := FALSE);
      State := 0;
    END_IF;

```

```
END_IF;
END_CASE;
```

### 10.3.2 Electronic gear unit with flying saw

For the electronic gear unit ("angularly synchronised operation") and the sub-function flying saw there are two function blocks which enable control of these functions. In addition, various parameters must be set for the correct execution of the two function blocks in the master and slave frequency inverters. An example of this is shown in the following table (explained by the example of a SK 540E).

Master FI			Slave FI		
Parameter	Settings	Description	Parameter	Settings	Description
P502[-01]	20	Setpoint frequency according to freq. Ramp	P509	10 *	CANopen Broadcast *
P502[-02]	15	Actual position in incl. High word	P510[-01]	10	CANopen Broadcast
P502[-03]	10	Actual position in incl. Low word	P510[-02]	10	CANopen Broadcast
P503	3	CANopen	P505	0	0,0 Hz
P505	0	0.0 Hz	P515[-02]	P515[-03] <sub>Master</sub>	Broadcast Slave address
P514	5	250 kBaud (min. 100 kBaud)	P546[-01]	4	Frequency addition
P515[-03]	P515[-02] Slave	Broadcast master address	P546[-02]	24	Setpoint pos. Incl. High Word
			P546[-03]	23	Setpoint pos. Incl. Low Word
			P600	1.2	Position control ON
			Only for FB_Gearing		
			P553[-01]	21	Position setpoint pos. Low word
			P553[-02]	22	Position setpoint pos. High word

\* (P509) must not necessarily be set to {10} "CANopen Broadcast". However, in this case the Master (P502 [-01]) must be set to {21} "Actual frequency without slip".

**i Information**

**Actual position – transmission format**

The actual position of the master MUST be communicated in "Increments" (Inc) format.

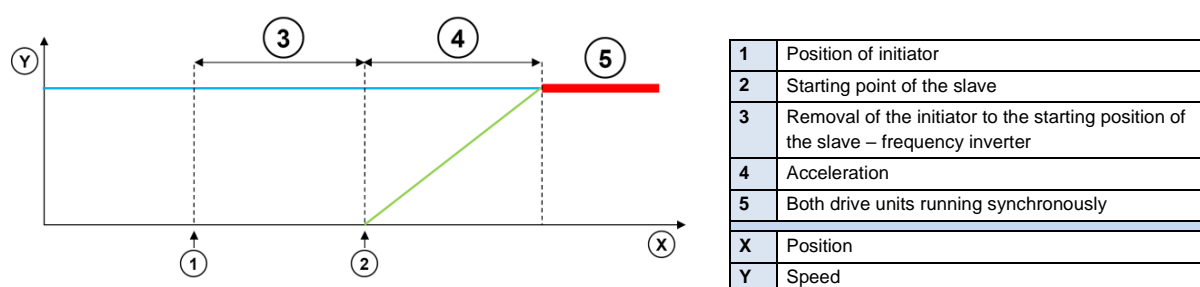
### 10.3.2.1 Overview

Function module	Description
FB_Gearing	FB for simple gear unit function
FB_FlyingSaw	FB for gear unit function with Flying Saw

### 10.3.2.2 FB\_FlyingSaw

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	

The flying saw function is an extension of the gear unit function. With the aid of this function it is possible to synchronise a running drive unit to a precise position. In contrast to FB\_Gearing, synchronisation is relative, i.e. the slave axis moves synchronously to the position of the master which applied at the start of the "Flying Saw". The synchronisation process is illustrated in the figure below.



If the function is started, the slave frequency inverter accelerates to the speed of the master axis. The acceleration is specified via **ACCELERATION**. At low speeds the ramp is flatter and at high speeds there is a steep ramp for the slave frequency inverter. The acceleration path is stated in revolutions (1000 = 1,000 rev.) if P553 is specified as the setpoint position. If the setpoint position INC is used for P553, the acceleration path is specified in increments.

If the initiator is set with the distance of the position of the slave drive which is saved in **ACCELERATION**, the slave is precisely synchronised with the triggering position from the master drive.

The FB must be switched on via the **ENABLE** input. The function can be started either via the digital input (P420[-xx]=64, Start Flying Saw) or via **EXECUTE**. The frequency inverter then accelerates to the speed of the master axis. When synchronisation with the master axis is achieved, the **DONE** output is switched to 1.

Via the **STOP** input or the digital input function P420[-xx] = 77, Stop Flying Saw, the gear unit function is switched off, the frequency inverter brakes to 0 Hz and remains at a standstill. Via the **HOME** input, the inverter is made to move to the absolute position 0. After the end of the **HOME** or **STOP** command the relevant allocated output is active. The gear unit function can be restarted with renewed activation of **EXECUTE** or the digital input. With the digital input function (P420[-xx] = 63, Stop synchronisation) the gear unit function can be stopped and then moved to the absolute position 0.

If the function is interrupted by the MC\_Stop function, **ABORT** is set to 1. In case of error, **ERROR** is set to 1 and the error code is set in **ERRORID**. These three outputs are reset if **ENABLE** is switched to 0.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Enable	BOOL	<b>VALID</b>	Specified setpoint frequency reached	BOOL
<b>EXECUTE</b>	Start of synchronisation	BOOL	<b>DONEHOME</b>	Home run completed	
<b>STOP</b>	Stop synchronisation	BOOL	<b>DONESTOP</b>	Stop command executed	
<b>HOME</b>	Moves to position 0	BOOL	<b>ABORT</b>	Command aborted	BOOL
<b>ACCELERATION</b>	Acceleration path (1rev. = 1.000)	DINT	<b>ERROR</b>	Error in FB	BOOL
			<b>ERRORID</b>	Error code	INT
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1000h	FI is not enabled				
1200h	Position control not activated				

### 10.3.2.3 FB\_Gearing

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	

The position and speed of the frequency inverter can be synchronised to that of a master inverter via the function module FB\_Gearing. The slave which is used this function always follows the movements of the master inverter. Synchronisation is absolute, i.e. the positions of the slave and the master are always the same.

#### Information

If the slave is switched to gear unit mode at a different position to the master, the slave moves to the position of the master with the maximum frequency. If a gear ratio is specified, this also results in a new position when switched on again.

The position value to which synchronisation is carried out, as well as the speed, must be communicated via the Broadcast channel. The function is enabled via the input **ENABLE**. For this, the position control and the output stage must be enabled. The output stage can be enabled e.g. with the function MC\_Power. If **ENABLE** is set to 0, the frequency inverter brakes to 0 Hz and remains at a standstill. The inverter is now in position control mode again. If MC\_Stop is activated, the frequency inverter exits from the gear unit mode and the **ABORT** output changes to 1. In case of errors in the FB **ERROR** changes to 1 and the cause of the error is indicated in **ERRORID**. By setting **ENABLE** to 0, **ERROR**, **ERRORID** and **ABORT** can be reset.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Synchronous running active	BOOL	<b>VALID</b>	Gear unit function is active	BOOL
<b>RELATIVE</b>	Relative Mode (V2.1 and above)	BOOL	<b>ABORT</b>	Command aborted	BOOL
			<b>ERROR</b>	Error in FB	BOOL
			<b>ERRORID</b>	Error code	INT
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1000h	FI is not enabled				
1200h	Position control not activated				
1201h	The PLC setpoint position High is not parameterised				
1202h	The PLC setpoint position Low is not parameterised				

Gear ratios or a change of direction of rotation can be set via the parameters P607[-05] or P608[-05]. Further details can be found in Manual BU0510 (Supplementary manual for POSICON position control).

### 10.3.3 Motion Control

The Motion Control Lib is based on the PLCopen specification "Function blocks for motion control". It contains function blocks for controlling and moving a frequency inverter and provides access to its parameters. Several settings must be made to the parameters of the device in order for the Motion Blocks to function.

Function blocks	Required settings
MC_MoveVelocity	P350 = PLC active P351 = Main setpoint comes from the PLC P553 [-xx] = Setpoint frequency P600 = Position control (positioning mode) is disabled
MC_MoveAbsolute	P350 = PLC active
MC_MoveRelative	P351 = Main setpoint comes from the PLC
MC_MoveAdditive	P600 = Position control (positioning mode) is enabled
MC_Home	In P553 [-xx] ( PLC_Setpoints ) the setpoint position High Word must be parameterised In P553 [-xx] ( PLC_Setpoints ) the setpoint position Low Word must be parameterised In P553 [-xx] ( PLC_Setpoints ) the setpoint frequency must be parameterised
MC_Power	P350 = PLC active
MC_Reset	P351 = Control word comes from the PLC
MC_Stop	

**i Information**

The PLC\_Setpoints 1 to 5 and the PLC control word can also be described via process variables. However, if the Motion Control FBs are used, no corresponding process variable may be declared in the table of variables, as otherwise the outputs of the Motion Control FBs would be overwritten.

**i Information**
**Detecting a signal edge**

In order for the following function blocks to detect an edge at the input, it is necessary for the function call-up to be carried out twice with different statuses at the input.

Function blocks	Description
MC_ReadParameter	Reading access to parameters of the device
MC_WriteParameter	Writing access to parameters of the device
MC_MoveVelocity	Move command in speed mode
MC_MoveAbsolute	Move command with specification of absolute position
MC_MoveRelative	Move command with specification of relative position
MC_MoveAdditive	Move command with additive specification of position
MC_Home	Starts a home run
MC_Power	Switches the motor voltage on or off
MC_ReadStatus	Status of the device
MC_ReadActualPos	Reads out the actual position
MC_Reset	Error reset in the device
MC_Stop	Stops all active movement commands

**10.3.3.1 MC\_Control**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	

This FB is used to control the FI and offers the possibility of creating the FI control word in a more detailed form than MC\_Power. The FI is controlled via the inputs **ENABLE**, **DISABLEVOLTAGE** and **QUICKSTOP**. See the following table.

Inputs module			Frequency inverter behaviour
ENABLE	QUICKSTOP	DISABLEVOLTAGE	
High	Low	Low	The frequency inverter is switched on.
Low	Low	Low	The frequency inverter brakes to 0 Hz (P103) and then



			disconnects the motor from the voltage supply.
X	X	High	The frequency inverter is disconnected from the voltage supply immediately and the motor runs to a standstill without braking.
X	High	Low	The frequency inverter makes an emergency stop (P426) and then disconnects the voltage from the motor.

The active parameter set can be set via the input **PARASET**. If the output **STATUS** = 1, the FI is switched on and current is supplied to the motor.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Enable	BOOL	<b>STATUS</b>	Motor is supplied with current	BOOL
<b>DISABLEVOLTAGE</b>	Switch off voltage	BOOL	<b>ERROR</b>	Error in FB	BOOL
<b>QUICKSTOP</b>	Quick stop	BOOL	<b>ERRORID</b>	Error code	INT
<b>PARASET</b>	Active parameter set Value range: 0 – 3	BYTE			
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1001h	Stop function is active				
1300h	The FI is in an unexpected state				

### Example ST:

```
(* Device enabled with Dig3 *)
Control.Enable := _5_State_digital_input.2;
(* Parameter sets are specified via Dig1 and Dig2. *)
Control.ParaSet := INT_TO_BYTE(_5_State_digital_input and 2#11);
Control;
(* Is the device enabled? *)
if Control.Status then
  (* Should a different position be moved to? *)
  if SaveBit3 <> _5_State_digital_input.3 then
    SaveBit3 := _5_State_digital_input.3;
    if SaveBit3 then
      Move.Position := 500000;
    else
      Move.Position := 0;
    end_if;

    Move(Execute := False);
  end_if;
end_if;

(* Move to position if device is enabled. *)
Move(Execute := Control.Status);
```

### 10.3.3.2 MC\_Control\_MS

	<b>SK 54xE</b>	<b>SK 53xE</b>	<b>SK 2xxE</b>	<b>SK 2xxE-FDS</b>	<b>SK 180E</b>	<b>SK 155E-FDS</b>
--	----------------	----------------	----------------	--------------------	----------------	--------------------

		SK 52xE			SK 190E	SK 175E-FDS
Availability						X

This FB is used to control the starter (MS).

Inputs module				Frequency inverter behaviour
ENABLE_RIG HT	ENABLE_LEF T	QUICKSTOP	DISABLEVOLTAGE	
High	Low	Low	Low	The MS is switched on with rotation clockwise
Low	High	Low	Low	The MS is switched on with rotation anticlockwise
High	High	Low	Low	The MS is switched off
Low	Low	Low	Low	The MS brakes to 0 Hz (P103) and then disconnects the motor from the voltage supply
X	X	X	High	The MS is disconnected from the voltage supply immediately and the motor runs to a standstill without braking.
X	X	High	Low	The MS makes an emergency stop (P426) and then disconnects the voltage from the motor.

(X = The level at the input is irrelevant)

If the output **STATUS** = 1 the MS is switched on and current is supplied to the motor.

If **OPENBRAKE** is set to 1 the brake is released.

VAR_INPUT			VAR_OUTPUT		
Input	Explanation	Type	Output	Explanation	Type
<b>ENABLE_RIGHT</b>	Enable right	BOOL	<b>STATUS</b>	Motor is supplied with current	BOOL
<b>ENABLE_LEFT</b>	Enable left	BOOL	<b>ERROR</b>	Error in FB	BOOL
<b>DISABLEVOLTAGE</b>	Switch off voltage	BOOL	<b>ERRORID</b>	Error code	INT
<b>QUICKSTOP</b>	Quick stop	BOOL			
<b>OPENBRAKE</b>	Release brake	BOOL			
<b>ERRORID</b>	<b>Explanation</b>				
0	No error				
1001h	Stop function is active				
1300h	The MS is in an unexpected state				

### 10.3.3.3 MC\_Home

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	

Causes the frequency inverter to start a reference run, if **EXECUTE** changes from 0 to 1 (flank). The frequency inverter moves with the setpoint frequency which is entered in **VELOCITY**. The direction of rotation is reversed if the input with the position reference signal (P420[-xx] = Reference point) becomes active. With a negative flank of the position reference signal, the value in **POSITION** is adopted. The frequency inverter then brakes to 0 Hz and the **DONE** signal changes to 1. During the entire **HOME** run, the **BUSY** output is active. If the input is "MODE" is set to "True", the inverter runs in the middle of the initiator after homing.

If the process is to be aborted (e.g. by a different MC function module), **COMMANDABORTED** is set. In case of error, **ERROR** is set to 1 in this case, **DONE** is 0. The corresponding error code in **ERRORID** then applies.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Enable	BOOL	<b>DONE</b>	Specified setpoint position reached	BOOL
<b>POSITION</b>	Setpoint position	DINT	<b>COMMAND-ABORTED</b>	Command aborted	BOOL
<b>VELOCITY</b>	Setpoint frequency	INT	<b>ERROR</b>	Error in FB	BOOL
<b>MODE</b>	Home Mode (from V2.1)	BOOL	<b>ERRORID</b>	Error code	INT
			<b>BUSY</b>	Home run active	BOOL

ERRORID	Description
0	No error
1000h	FI is not enabled
1200h	Position control not activated
1201h	The High position has not been entered in the PLC setpoints (P553)
1202h	The Low position has not been entered in the PLC setpoints (P553)
1D00h	Absolute encoders are not supported

**Example ST:**

```
(* One digital input must be set as the reference point (23).
  The setpoint frequency and the setpoint position must be set in the PLC setpoint
  (553.x). POSICON must be enabled (P600) *)

(* Enable device with DIG1 *)
Power.Enable := _5_State_digital_input.0;
(* Position after the reference run *)
Home.Position := 5000;
(* Speed during the reference run 50% of the max. frequency (P105) *)
Home.Velocity := 16#2000;
(* Perform reference run when the device is switched on *)
Home.Execute := Power.Status;

Home;
Power;
```

**10.3.3.4 MC\_MoveAbsolute**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	

Writes a position and speed setpoint to the frequency inverter if **EXECUTE** changes from 0 to 1 (flank). The setpoint frequency **VELOCITY** is transferred according to the scaling explained in MC\_MoveVelocity.

**POSITION:**

**MODE** = False:

The setpoint position results from the value transferred into **POSITION**.

**MODE** = True:

The value transferred into **POSITION** corresponds to the index from parameter P613 increased by 1. The position saved in this parameter index corresponds to the setpoint position.

Example:

Mode = True; Position = 12

The FB moves to the position which is in the current parameter set of P613[-13].

If the inverter has reached the setpoint position **DONE** is set to 1. **DONE** is deleted by resetting **EXECUTE**. If **EXECUTE** is deleted before the target position is reached, **DONE** is set to 1 for one cycle. During movement to the setpoint position **BUSY** is active. If the process is to be aborted (e.g. by

another MC function module), **COMMANDABORTED** is set. In case of error, **ERROR** is set to 1 and the corresponding error code is set in **ERRORID**. **DONE** is 0 in this case. With a negative flank on **EXECUTE**, all outputs are reset to 0.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Enable	BOOL	<b>DONE</b>	Specified setpoint position reached	BOOL
<b>POSITION</b>	Setpoint position	DINT	<b>BUSY</b>	Setpoint position not reached	BOOL
<b>VELOCITY</b>	Setpoint frequency	INT	<b>COMMAND-ABORTED</b>	Command aborted	BOOL
<b>MODE</b>	Mode source is the setpoint position	BOOL	<b>ERROR</b>	Error in FB	BOOL
			<b>ERRORID</b>	Error code	INT
<b>ERRORID</b>	<b>Description</b>				
0	No erro				
0x1000	FI is not enabled				
0x1200	Position control not activated				
0x1201	The High position has not been entered in the PLC setpoints (P553)				
0x1202	The Low position has not been entered in the PLC setpoints (P553)				

#### Example ST:

```
(* The device is enabled if DIG1 = TRUE *)
Power(Enable := _5_State_digital_input.0);
IF Power.Status THEN
  (* The device is enabled and moves to position 20000 with 50% max. frequency.
  For this action the motor requires an encoder and position control must be enabled. *)
  MoveAbs(Execute := _5_State_digital_input.1, Velocity := 16#2000, Position := 20000);
END_IF
```

#### 10.3.3.5 MC\_MoveAdditive

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	

Except for the input **DISTANCE** this corresponds in all points with MC\_MoveAbsolute. The setpoint position results from the addition of the actual setpoint position and the transferred **DISTANCE**.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Enable	BOOL	<b>DONE</b>	Specified setpoint position reached	BOOL
<b>DISTANCE</b>	Setpoint position	DINT	<b>COMMAND-</b>	Command aborted	BOOL

			<b>ABORTED</b>		
<b>VELOCITY</b>	Setpoint frequency	INT	<b>ERROR</b>	Error in FB	BOOL
<b>MODE</b>	Mode source is the setpoint position	BOOL	<b>ERRORID</b>	Error code	INT
			<b>BUSY</b>	Setpoint position not reached	BOOL
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1000h	FI is not enabled				
1200h	Position control not activated				
1201h	The High position has not been entered in the PLC setpoints (P553)				
1202h	The Low position has not been entered in the PLC setpoints (P553)				

### 10.3.3.6 MC\_MoveRelative

	<b>SK 54xE</b>	<b>SK 53xE SK 52xE</b>	<b>SK 2xxE</b>	<b>SK 2xxE-FDS</b>	<b>SK 180E SK 190E</b>	<b>SK 155E-FDS SK 175E-FDS</b>
<b>Availability</b>	X	X	X	X	X	

Except for the input **DISTANCE** this corresponds in all points with MC\_MoveAbsolute. The setpoint position results from the addition of the current actual position and the transferred **DISTANCE**.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Enable	BOOL	<b>DONE</b>	Specified setpoint position reached	BOOL
<b>DISTANCE</b>	Setpoint position	DINT	<b>COMMAND-ABORTED</b>	Command aborted	BOOL
<b>VELOCITY</b>	Setpoint frequency	INT	<b>ERROR</b>	Error in FB	BOOL
<b>MODE</b>	Mode source is the setpoint position	BOOL	<b>ERRORID</b>	Error code	INT
			<b>BUSY</b>	Setpoint position not reached	BOOL
<b>ERRORID</b>	<b>Description</b>				
0	No erro				
1000h	FI is not enabled				
1200h	Position control not activated				
1201h	The High position has not been entered in the PLC setpoints (P553)				
1202h	The Low position has not been entered in the PLC setpoints (P553)				

### 10.3.3.7 MC\_MoveVelocity

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	

Sets the setpoint frequency for the frequency inverter if **EXECUTE** changes from 0 to 1 (flank). If the frequency inverter has reached the setpoint frequency, **INVELOCITY** is set to 1. While the FI is accelerating to the setpoint frequency, the **BUSY** output is active. If **EXECUTE** has already been set to 0, then **INVELOCITY** is only set to 1 for one cycle. If the process is to be aborted (e.g. by another MC function module), **COMMANDABORTED** is set.

With a negative flank on **EXECUTE**, all outputs are reset to 0.

**VELOCITY** is entered with scaling according to the following formula:

$$\text{VELOCITY} = (\text{Setpoint frequency (Hz)} \times 0x4000) / P105$$

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Erläuterung	Type
<b>EXECUTE</b>	Enable	BOOL	<b>INVELOCITY</b>	Specified setpoint frequency reached	BOOL
<b>VELOCITY</b>	Setpoint frequency	INT	<b>BUSY</b>	Setpoint frequency not yet reached	BOOL
			<b>COMMAND-ABORTED</b>	Command aborted	BOOL
			<b>ERROR</b>	Error in FB	BOOL
			<b>ERRORID</b>	Error code	INT
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1000h	FI is not enabled				
1100h	FI not in speed mode (position control enabled)				
1101h	No setpoint frequency parameterized (P553)				

#### Example AWL:

```

CAL Power
CAL Move

LD TRUE
ST Power.Enable

(* Set 20 Hz (Max. 50 Hz) *)
LD DINT#20
MUL 16#4000
DIV 50

DINT_TO_INT
ST Move.Velocity

LD Power.Status
ST Move.Execute

```

**Example ST:**

```
(* Device ready for operation if DIG1 set *)
Power(Enable := _5_State_digital_input.0);
IF Power.Status THEN
  (* Device enabled with 50% of max. frequency if DIG2 set *)
  MoveVelocity(Execute := _5_State_digital_input.1, Velocity := 16#2000);
END_IF
```

**10.3.3.8 MC\_Power**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

The output stage of the device can be switched on and off with this function. If the **ENABLE** input is set to 1, the output stage is enabled. The condition for this is that the device is in the state "Switch-on Block" or "Ready for Switch-on. If the FI is in the "Fault" or "Fault reaction active" state, the fault must first be remedied and acknowledged. Only then can enabling be carried out via this block. If the device is in the state "Not Ready for Switch-on", switch-on is not possible. In all cases, the FB goes into the error state and **ENABLE** must be set to 0 to acknowledge the fault.

If the **ENABLE** input is set to 0, the device is switched off. If this happens while the motor is running, it is first run down to 0 Hz via the ramp set in P103.

The output **STATUS** is 1 if the output stage of the device is switched on; otherwise it is 0.

**ERROR** and **ERRORID** are reset, if **ENABLE** is switched to 0.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Enable	BOOL	<b>STATUS</b>	Motor is supplied with current	BOOL
			<b>ERROR</b>	Error in FB	BOOL
			<b>ERRORID</b>	Error code	INT
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1001h	Stop function is active				
1300h	The device is not in the state "Ready for Switch-on" or "Switch-on Block"				



**Example AWL:**

```

CAL Power
CAL Move

LD TRUE
ST Power.Enable

(* (* Set 20 Hz (Max. 50 Hz) *) *)
LD DINT#20
MUL 16#4000
DIV 50

DINT_TO_INT
ST Move.Velocity

LD Power.Status
ST Move.Execute

```

**Example ST:**

```

(* Enable Power Block *)
Power(Enable := TRUE);
IF Power.Status THEN
  (* The device is ready for switch-on *)
END_IF

```

**10.3.3.9 MC\_ReadActualPos**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	

Continually delivers the actual position of the frequency inverter if **ENABLE** is set to 1. As soon as there is a valid position at the output **VALID** is set to valid. In case of error, **ERROR** is set to 1 and in this case **VALID** is 0.

Position scaling: 1 motor revolution = 1000

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Ausgang	Description	Type
<b>ENABLE</b>	Enable	BOOL	<b>VALID</b>	Output is valid	BOOL
			<b>ERROR</b>	Error in FB	BOOL
			<b>POSITION</b>	Current position	DINT

**Example ST:**

```

ReadActualPos(Enable := TRUE);
IF ReadActualPos.Valid THEN
  Pos := ReadActualPos.Position;
END_IF

```

**10.3.3.10 MC\_ReadParameter**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

Reads out a parameter cyclically from the device as long as **ENABLE** is set to 1. The parameter which is read is saved in Value and is valid if **DONE** is set to 1. For the duration of the reading process the **BUSY** output is set to 1. If **ENABLE** remains set to 1, the parameter is read out cyclically. The parameter number and index can be changed at any time when **ENABLE** is active. However, it is difficult to identify when the new value is read out, as the **DONE** signal remains 1 for the whole time. In this case it is advisable to set the **ENABLE** signal to 0 for one cycle, as the **DONE** signal is then reset. The parameter index results from the index in the documentation minus 1. E.g. P700 Index 3 ("Reason for switch-on block") is queried via the parameter index 2. In case of error **ERROR** is set to 1. **DONE** in this case is 0 and the **ERRORID** contains the error code. If the **ENABLE** signal is set to 0, all signals and the **ERRORID** are deleted.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Typ
<b>ENABLE</b>	Enable	BOOL	<b>DONE</b>	Value is valid	BOOL
<b>PARAMETERNUMBER</b>	Parameter number	INT	<b>ERROR</b>	Reading has failed	BOOL
<b>PARAMETERINDEX</b>	Parameter index	INT	<b>BUSY</b>	The process is not complete	BOOL
			<b>ERRORID</b>	Error code	INT
			<b>VALUE</b>	Parameter read out	DINT
<b>ERRORID</b>	<b>Description</b>				
0	Invalid parameter number				
3	Incorrect parameter index				
4	No array				
201	Invalid order element in the last order received				
202	Internal response label cannot be depicted				

#### Example ST:

```
(* Motion module FB_ReadParameter *)
ReadParam(Enable := TRUE, Parameternumber := 102, ParameterIndex := 0);
IF ReadParam.Done THEN
    Value := ReadParam.Value;
    ReadParam(Enable := FALSE);
END_IF
```

#### 10.3.3.11 MC\_ReadStatus

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

Reads out the status of the device. The status machine is orientated to the PLCopen specification "Function blocks for motion control". The status is read out as long as **ENABLE** is set to 1.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Enable	BOOL	<b>VALID</b>	Output is valid	BOOL
			<b>ERROR</b>	Error in FB	BOOL
			<b>ERRORSTOP</b>	The device has an error	BOOL
			<b>DISABLED</b>	The output stage of the device is switched off	BOOL
			<b>STOPPING</b>	A Stop command is active	BOOL
			<b>DISCRETEMOTION</b>	One of the three positioning FBs is active	BOOL
			<b>CONTINUOUSMOTION</b>	The MC_Velocity is active	BOOL
			<b>HOMING</b>	The MC_Home is active	BOOL
			<b>STANDSTILL</b>	The device has no active move command. It is at a standstill with 0 rpm and the output stage switched on.	BOOL

**Example ST:**

```

ReadStatus(Enable := TRUE);
IF ReadStatus.Valid THEN
  fError := ReadStatus.ErrorStop;
  fDisable := ReadStatus.Disabled;
  fStopping := ReadStatus.Stopping;
  fInMotion := ReadStatus.DiscreteMotion;
  fInVelocity := ReadStatus.ContinuousMotion;
  fInHome := ReadStatus.Homing;
  fStandStill := ReadStatus.StandStill;
end_if

```

**10.3.3.12 MC\_Reset**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

Resets an error in the device (fault acknowledgement), on a rising flank from **EXECUTE**. In case of error **ERROR** is set to 1 and the cause of the fault is entered in **ERRORID**. With a negative flank on **EXECUTE** all errors are reset.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Start	BOOL	<b>DONE</b>	device error reset	BOOL
			<b>ERROR</b>	Error in FB	BOOL

			<b>ERRORID</b>	Error code	INT
			<b>BUSY</b>	Reset process is still active	BOOL
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1001h	Stop function is active				
1700h	An error reset could not be performed, because the cause of the error is still present.				

**Example ST:**

```

Reset(Execute := TRUE);
IF Reset.Done THEN
  (* The error has been reset *)
  Reset(Execute := FALSE);
ELSIF Reset.Error THEN
  (* Reset could not be executed, as the cause of the error is still present *)
  Reset(Execute := FALSE);
END_IF

```

**10.3.3.13 MC\_Stop**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

With a rising flank (0 to 1) the device is set to the state **STANDINGSTILL**. All motion functions which are active are cancelled. The device brakes to 0 Hz and switches off the output stage. As long as the Stop command is active (**EXECUTE** = 1), all other Motion FBs are blocked. The **BUSY** output becomes active with the rising flank on **EXECUTE** and remains active until there is a falling flank on **EXECUTE**.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Start	BOOL	<b>DONE</b>	Command has been executed	BOOL
			<b>BUSY</b>	Command is active	BOOL

**10.3.3.14 MC\_WriteParameter\_16 / MC\_WriteParameter\_32**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

Writes a 16/32 Bit parameter into the device if **EXECUTE** changes from 0 to 1 (flank). The parameter has been written if **DONE** is set to 1. For the duration of the reading process the **BUSY** output is set to 1. In case of error, **ERROR** is set to 1 and the **ERRORID** contains the error code. The signals **DONE**, **ERROR**, **ERRORID** remain set until **EXECUTE** changes back to 0. If the **EXECUTE** signal changes to 0, the writing process is not cancelled. Only the **DONE** signal remains set for 1 PLC cycle. If the input

**RAMONLY** is set to 1, then the value is only stored RAM. The changed settings are lost when you turn off the device.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Enable	BOOL	<b>DONE</b>	Value is valid	BOOL
<b>PARAMETERNUMBER</b>	Parameter number	INT	<b>BUSY</b>	The writing process is active	BOOL
<b>PARAMETERINDEX</b>	Parameter index	INT	<b>ERROR</b>	Reading has failed	BOOL
<b>VALUE</b>	Value to be written	INT	<b>ERRORID</b>	Error code	INT
<b>RAMONLY</b>	Store the value only in RAM (from V2.1)	BOOL			
<b>ERRORID</b>	<b>Description</b>				
0	Invalid parameter number				
1	Parameter value cannot be changed				
2	Lower or upper value limit exceeded				
3	Incorrect parameter index				
4	No array				
5	Invalid data type				
6	Only resettable (only 0 may be written)				
7	Description element cannot be changed				
201	Invalid order element in the last order received				
202	Internal response label cannot be depicted				

### Example ST:

```

WriteParam16(Execute := TRUE, ParameterNumber := 102, ParameterIndex := 0, Value := 300);
IF WriteParam16.Done THEN
  WriteParam16(Execute := FALSE);
END_IF;

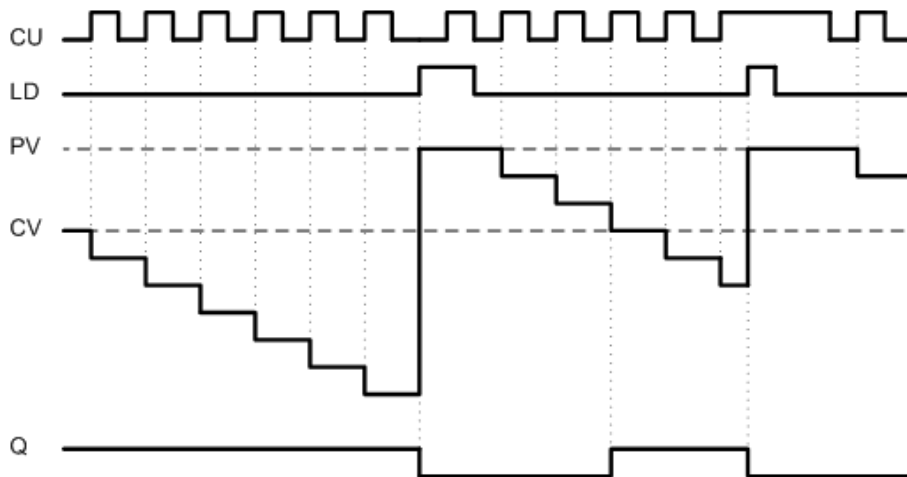
```

## 10.3.4 Standard

### 10.3.4.1 CTD downward counter

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

With a rising flank on CD the counter of the function block CV is reduced by one, as long as CV is larger than -32768. If CV is less than or equal to 0, the output Q remains TRUE. Via LD the counter CV can be set to the value saved in PV.



VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
CD	Counter input	BOOL	Q	TRUE, if CV ≤ 0	BOOL
LD	Load starting value	BOOL	CV	Actual counter reading	INT
PV	Starting value	INT			

**Example AWL:**

```
LD VarBOOL1
ST CTDInst.CD
LD VarBOOL2
ST CTDInst.LD
LD VarINT1
ST CTDInst.PV
CAL CTDInst
LD CTDInst.Q
ST VarBOOL3
LD CTDInst.CV
ST VarINT2
```

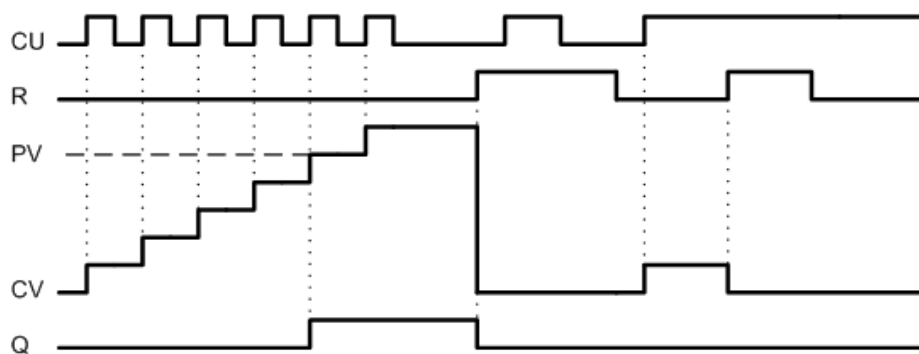
**Example ST:**

```
CTDInst(CD := VarBOOL1, LD := VarBOOL2, PV := VarINT1);
VarBOOL3 := CTDInst.Q;
VarINT2 := CTDInst.CV;
```

**10.3.4.2 CTU upward counter**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

With a rising flank on CU, the counter of the function block CV is increased by one. CV can be counted up to the value 32767. As long as CV is greater than or equal to PV, output Q remains TRUE. Via R the counter CV can be reset to zero.



VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
CU	Counter input	BOOL	Q	TRUE, if CV >= PV	BOOL
R	Reset: counter reading	BOOL	CV	Actual counter reading	INT
PV	Max. counter value	INT			

**Example AWL:**

```
LD VarBOOL1
ST CTUInst.CU
LD VarBOOL2
ST CTUInst.R
LD VarINT1
ST CTUInst.PV
CAL CTUInst
LD CTUInst.Q
ST VarBOOL3
LD CTUInst.CV
ST VarINT2
```

**Example ST:**

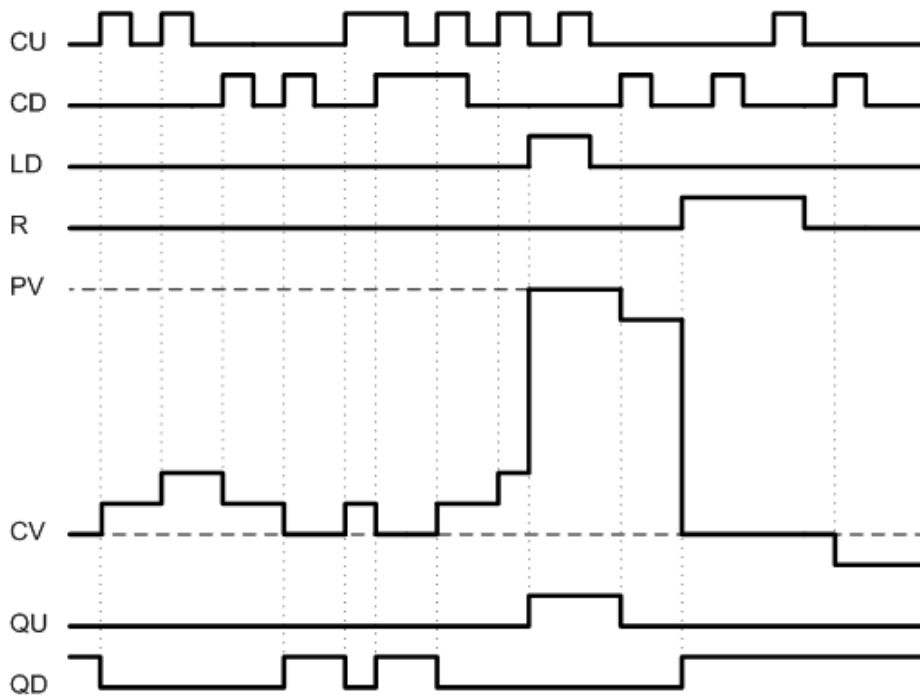
```
CTUInst(CU := VarBOOL1, R := VarBOOL2, PV := VarINT1);
VarBOOL3 := CTUInst.Q;
VarINT2 := CTUInst.CV;
```

**10.3.4.3 CTUD upward and downward counter**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

With a rising flank on CU the counter CV is increased by one, as long as CV is less than 32767. With a rising flank on CD the counter CV is reduced by one, as long as CV is greater than -32768. Via R the counter CV can be set to zero. Via LD the value saved in PV is copied to CV.

R have priority over LD, CU and CV. PV can be changed at any time, QU always relates to the value which is currently set.



VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>CU</b>	Counting upwards	BOOL	<b>QU</b>	TRUE, if CV >= PV	BOOL
<b>CD</b>	Counting downwards	BOOL	<b>QD</b>	TRUE, if CV <= 0	BOOL
<b>R</b>	Reset: counter reading	BOOL	<b>CV</b>	Actual counter reading	INT
<b>LD</b>	Load starting value	BOOL			
<b>PV</b>	Starting value	INT			

**Example AWL:**

```

LD VarBOOL1
ST CTUDInst.CU
LD VarBOOL3
ST CTUDInst.R
LD VarBool4
ST CTUDInst.LD
LD VarINT1
ST CTUInst.PV
CAL CTUDInst
LD CTUDInst.Q
ST VarBOOL5
LD CTUDInst.QD
ST VarBOOL5
LD CTUInst.CV
ST VarINT2
    
```



**Example ST:**

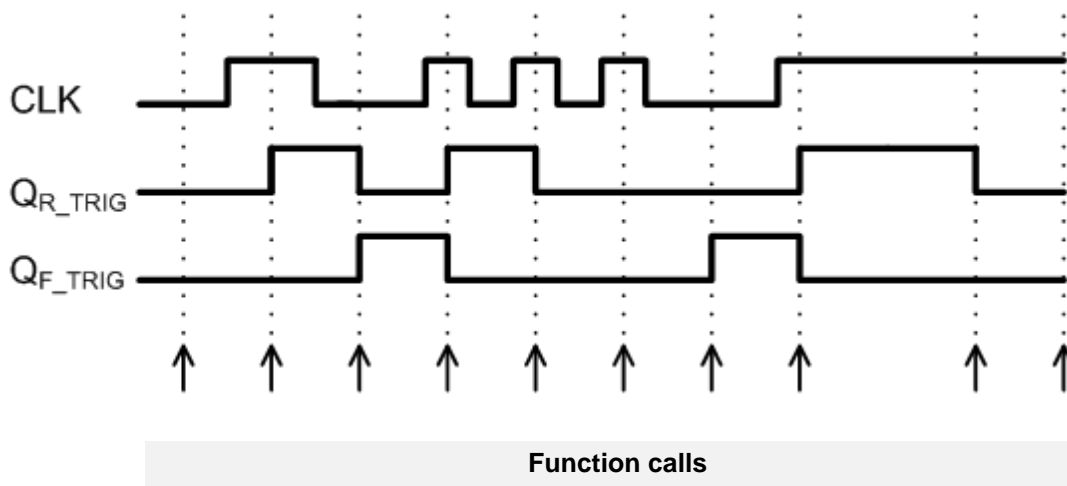
```
CTUDInst (CU:=VarBOOL1, R:=VarBOOL3, LD:=VarBOOL4, PV:=VarINT1);
VarBOOL5 := CTUDInst.QU;
VarBOOL5 := CTUDInst.QD;
VarINT2 := CTUDInst.CV;
```

**10.3.4.4 R\_TRIG und F\_TRIG**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

Both functions are used for flank detection. If a flank is detected on CLK, Q is set to TRUE until the next function call-up, after which it is reset to FALSE. Only with a new flank can Q become TRUE again for a cycle.

- R\_TRIG = Rising flank
- F\_TRIG = Falling flank



VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
CLK	Set	BOOL	Q	Output	BOOL

**Example in AWL:**

```
LD VarBOOL1
ST RTRIGInst.CLK
CAL RTRIGInst
LD RTRIGInst.Q
ST VarBOOL2
```

**Example in ST:**

```
RTRIGInst (CLK:= VarBOOL1);
VarBOOL2 := RTRIGInst.Q;
```

**i Information**

The output of the function only changes if the function is called up. Because of this it is advisable to continually call up flank detection with the PLC cycle.

**10.3.4.5 RS Flip Flop**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X

Bi-stable function: via **S** the output **Q1** is set and via **R1** it is deleted again. If **R1** and **S** are both TRUE, **R1** is dominant.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>S</b>	Set	BOOL	<b>Q1</b>	Output	BOOL
<b>R1</b>	Reset	BOOL			

**Example in AWL:**

```
LD VarBOOL1
ST RSInst.S
LD VarBOOL2
ST RSInst.R1
CAL RSInst
LD RSInst.Q1
ST VarBOOL3
```

**Example in ST:**

```
RSInst(S:= VarBOOL1 , R1:=VarBOOL2);
VarBOOL3 := RSInst.Q1;
```

**10.3.4.6 SR Flip Flop**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X

Bi-stable function; via **S1** the output **Q1** is set and via **R** it is deleted again. If **R** and **S1** are both TRUE, **S1** is dominant.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>S1</b>	Set	BOOL	<b>Q1</b>	Output	BOOL
<b>R</b>	Reset	BOOL			

**Example AWL:**

```
LD VarBOOL1
ST SRInst.S1
LD VarBOOL2
ST SRInst.R
CAL RSInst
LD SRInst.Q1
ST VarBOOL3
```

**Example ST:**

```
SRInst(S1:= VarBOOL1 , R:=VarBOOL2);
VarBOOL3 := SRInst.Q1;
```

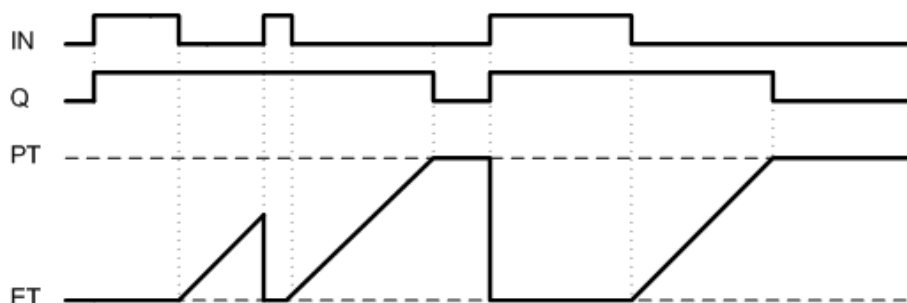
**10.3.4.7 TOF switch-off delay**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

If IN = TRUE, then Q is set to TRUE. If IN changes to FALSE, the timer counts upwards. As long as the timer is running (ET < PT) Q remains set to TRUE. If (ET = PT) the timer stops and Q becomes FALSE. With a new rising flank on IN, the timer ET is reset to zero.

Here, literals can be used for simplified input, e.g.

- LD TIME#50s20ms = 50.020 seconds
- LD TIME#1d30m = 1 day and 30 minutes



VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
IN	Timer active	BOOL	Q	TRUE & ( ET < PT )	BOOL
PT	Duration	DINT	ET	Current timer reading	DINT

**Example AWL:**

```
LD VarBOOL1
ST TOFInst.IN
LD DINT#5000
ST TOFInst.PT
CAL TOFInst
LD TOFInst.Q
ST VarBOOL2
```

**Example ST:**

```
TOFInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 := TOFInst.Q;
```

**i Information**

**Timer ET**

The time ET runs independently of a PLC cycle. Starting of the timer with IN and setting of the output Q are only executed with the function call-up "CAL". The function call-up takes place within a PLC cycle. However, with PLC programs which are longer than 5 ms this may result in the occurrence of jitter.

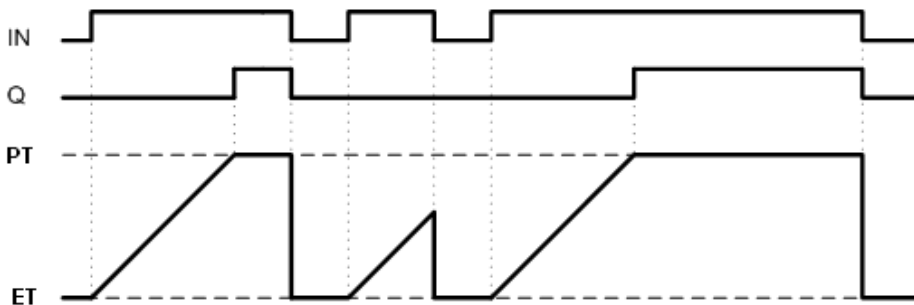
**10.3.4.8 TON switch-on delay**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

If IN = TRUE is set, the timer counts upwards. If ET = PT, Q is set to TRUE and the timer stops. Q remains TRUE for as long as IN is also TRUE. With a new rising flank on IN the counter starts again from zero. PT can be changed while the timer is running. The time period in PT in is entered in milliseconds. This enables a time delay between 5 ms and 24.8 days. As the time base of the PLC is 5 ms, the minimum time delay is also 5 ms.

Here, literals can be used for simplified input, e.g.

- LD TIME#50s20ms = 50.020 seconds
- LD TIME#1d30m = 1 day and 30 minutes



VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
IN	Timer active	BOOL	Q	TRUE $\beta$ ( IN=TRUE & ET=PT )	BOOL
PT	Duration	DINT	ET	Current timer reading	DINT

**Example AWL:**

```
LD VarBOOL1
ST TONInst.IN
LD DINT#5000
ST TONInst.PT
CAL TONInst
LD TONInst.Q
ST VarBOOL2
```

**Example ST:**

```
TONInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 := TONInst.Q;
```

**Information** **Timer ET**

The time ET runs independently of a PLC cycle. Starting of the timer with IN and setting of the output Q are only executed with the function call-up "CAL". The function call-up takes place within a PLC cycle. However, with PLC programs which are longer than 5 ms this may result in the occurrence of jitter.

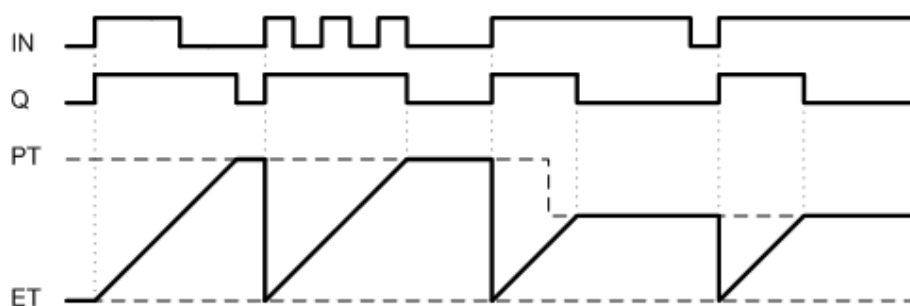
**10.3.4.9 TP time pulse**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

With a positive flank on IN the timer is started with the value 0. The timer runs up to the value which is entered PT and then stops. This process cannot be interrupted! PT can be changed during counting. The output Q is TRUE, as long as the timer ET is less than PT. If ET = PT and a rising flank is detected on IN the timer is started again at 0.

Here, literals can be used for simplified input, e.g.

- LD TIME#50s20ms = 50.020 seconds
- LD TIME#1d30m = 1 day and 30 minutes



VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
IN	Timer active	BOOL	Q	TRUE β ( ET < PT )	BOOL
PT	Duration	DINT	ET	Current timer reading	DINT

**Example AWL:**

```
LD VarBOOL1
ST TPInst.IN
LD DINT#5000
ST TPInst.PT
CAL TPInst
LD TPInst.Q
ST VarBOOL2
```

**Example ST:**

```
TPInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 := TPInst.Q;
```

**i Information**

**Timer ET**

The time ET runs independently of a PLC cycle. Starting of the timer with IN and setting of the output Q are only executed with the function call-up "CAL". The function call-up takes place within a PLC cycle. However, with PLC programs which are longer than 5 ms this may result in the occurrence of jitter.

**10.3.5 Access to memory areas of the frequency inverter**

If the intermediate saving of large quantities of data, its transmission to or reception from other devices is necessary, the modules FB\_WriteTrace and FB\_ReadTrace should be used.

**10.3.5.1 FB\_ReadTrace**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X		

The memory areas of the FI can be read out directly with the aid of this FB.

If the FB detects a positive flank on ENABLE, all parameters with are present on the input are adopted. The memory address which is to be read out is indicated with STARTINDEX and MEMORY. If the reading process is successful the VALID output changes to 1 and the value which has been read out is in VALUE.

If the FB is now called up several times and the ENABLE input remains at 1, with each call up the memory address which is to be read out is increased by 1 and the content of the new memory address is immediately copied to the output VALUE.

The current memory index for the next access can be read out under the output ACTINDEX. If the end of the memory has been reached, the READY changes to 1 and the reading process is stopped.

Values can be read in INT or DINT format. For INT values, only the Low component is evaluated by the VALUE output. Allocation is carried out via the SIZE input; a 0 stands for INT and a 1 for DINT values.

Allocation of the memory areas is carried out via the MEMORY input:

**MEMORY = 1 à P613[0-251]** corresponds to 504 INT or 252 DINT values

**MEMORY = 0 à P900[0-247] to P906[0-111]** corresponds to 3200 INT or 1600 DINT values

The FB cannot be interrupted by other blocks. With a negative flank on ENABLE, all outputs are set to 0 and the function of the FB is terminated.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Execute	BOOL	<b>VALID</b>	Reading process successful	BOOL
<b>SIZE</b>	Memory format	BOOL	<b>READY</b>	The entire memory has been read out	BOOL
<b>MEMORY</b>	Selection of memory area	BYTE	<b>ERROR</b>	the FB has an error	BOOL
<b>STARTINDEX</b>	Indicates the memory cell to be written to	INT	<b>ERRORID</b>	Error code	INT
			<b>ACTINDEX</b>	Actual memory index, to which will be read in the next cycle	INT
			<b>VALUE</b>	Value read out	DINT
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1A00h	STARTINDEX value range exceeded				
1A01h	MEMORY value range exceeded				

### 10.3.5.2 FB\_WriteTrace

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X		

Via this FB, individual values or large numbers of values can be intermediately saved in the FI. The values are not permanently saved, i.e. the values are lost if the FI is restarted.

If the FB detects a positive flank on ENABLE, all parameters with are present on the input are adopted. The value in VALUE is written to the storage address indicated in STARTINDEX and MEMORY. If the writing process is successful, the VALID output changes to 1.

If the FB is now called up several times and the ENABLE input remains at 1, then with each call up of the FB the input VALUE is read and saved and the memory address is increased by 1. The current memory index for the next access can be read out under the output ACTINDEX. If the end of the memory is reached, the output FULL changes to 1 and the saving process is stopped. However, if the input OVERWRITE is set to 1, the memory index is reset to the STARTINDEX and the values which have been previously written are overwritten.

Values can be saved in INT or DINT format. For INT values, only the Low component is evaluated by the VALUE input. Allocation is carried out via the SIZE input; a 0 stands for INT and a 1 for DINT values.

The allocation of memory areas is carried out via the MEMORY input:

**MEMORY** = 1 à P613[0-251] corresponds to 504 INT or 252 DINT values

**MEMORY** = 0 à P900[0-247] to P906[0-111] corresponds to 3200 INT or 1600 DINT values

The FB cannot be interrupted by other blocks. With a negative flank on ENABLE all outputs are set to 0 and the function of the FB is ended.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Execute	BOOL	<b>VALID</b>	Writing process successful	BOOL
<b>SIZE</b>	Memory format	BOOL	<b>FULL</b>	Entire memory is full	BOOL
<b>OVERWRITE</b>	Memory can be overwritten	BOOL	<b>ERROR</b>	the FB has an error	BOOL
<b>MEMORY</b>	Selection of memory area	BYTE	<b>ERRORID</b>	Error code	INT
<b>STARTINDEX</b>	Indicates the memory cell to be written to	INT	<b>ACTINDEX</b>	Actual memory index, to which saving will be carried out in the next cycle	DINT
<b>VALUE</b>	Value to be saved	DINT			
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1A00h	STARTINDEX value range exceeded				
1A01h	MEMORY value range exceeded				

### Information


**Note!** The memory area in the setting MEMORY = 0 is also used by the Scope function. Use of the Scope function overwrites the saved values!

#### 10.3.6 Visualisation with ParameterBox

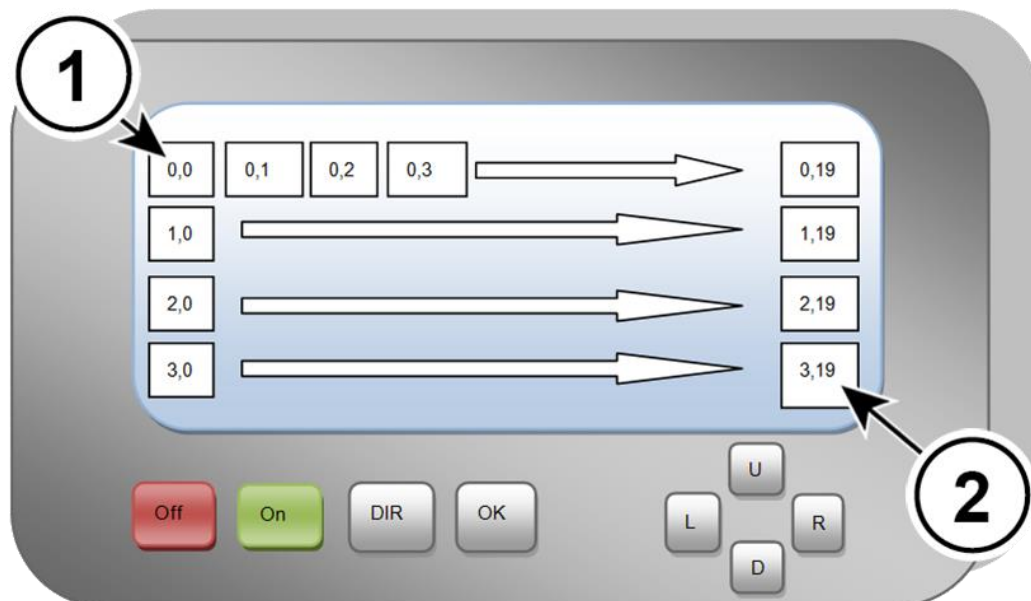
In the ParameterBox, the entire display can be used for the display of information. For this, the ParameterBox must be switched to visualisation mode. This is possible with the ParameterBox (Parameter P1308) firmware version V4.3 or higher, and is carried out as follows:

- In the menu item "Display", set the parameter P1003 to "PLC Display"
- Switch to the operating value display with the left or right arrow key
- PLC display is now enabled in the ParameterBox and remains permanently enabled.

In the visualisation mode of the ParameterBox, the content of the display can be set with the two FBs described below. However, before the item "Allow ParameterBox function modules" must be activated

in the PLC configuration dialogue (Button  ). With the process value "Parameterbox\_key\_state", the keyboard status of the box can also be queried. With this, input into the PLC program can be implemented. The display structure and the keys to be read out for the ParameterBox can be seen in the figure below.





1	First character	(0,0 → row = 0 , column = 0)
2	Last character	(3,19 → row = 3 , column = 19)

### 10.3.6.1 Overview visualisation

Function module	Description
FB_STRINGToPBox	Copies a string into the P-Box
FB_DINTToPBox	Copies a DINT value to the P-Box

### 10.3.6.2 FB\_DINTToPBOX

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

This function module converts a DINT value into an ASCII string and copies this into the ParameterBox. The output can be in decimal, binary or hexadecimal format; the selection is performed via MODE. Via ROW and COLUMN the starting point of the string is set in the ParameterBox display. The parameter LENGTH transfers the length of the string in characters. In decimal MODE the parameter POINT positions a decimal point in the number which is to be displayed. In POINT it is stated how many characters are to the right of the decimal point. With the setting 0 the POINT function is disabled. If the number contains more characters than the length allows and no decimal point is set, the overflow is indicated by the character "#". If there is a decimal point in the number, all numbers behind the decimal point may be omitted if required. In hexadecimal and binary MODE the lowest value bits are displayed if the set length is too short. As long as ENABLE is set to 1, all changes to the inputs are adopted immediately. If VALID changes to 1, the string has been correctly transferred. In case of error ERROR is set to 1. VALID is 0 in this case. In the ERRORID the relevant error code is then valid. With a negative flank on ENABLE, VALID, ERROR and ERRORID are reset.

#### Examples:

Setting	Number to be displayed	P-Box display
Length = 5	12345	12345
Point = 0		
Length = 5	-12345	#####
Point = 0		
Length = 10	123456789	123456,789
Point = 3		
Length = 8	123456789	123456,7
Point = 3		

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Transfer of the value	BOOL	<b>VALID</b>	Value transferred	BOOL
<b>MODE</b>	Display format 0 = Decimal 1 = Binary 2 = Hexadecimal Value range = 0 to 2	BYTE	<b>ERROR</b>	Error in FB	BOOL
<b>ROW</b>	Line of the display Value range = 0 to 3	BYTE	<b>ERRORID</b>	Error code	INT
<b>COLUMN</b>	Column of the display Value range = 0 to 19	BYTE			
<b>POINT</b>	Position of decimal point Value range = 0 to 10 0 = Function is disabled	BYTE			
<b>LENGTH</b>	Output length Value range = 1 to 11	BYTE			
<b>VALUE</b>	Number to be output	DINT			
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1500h	String overwrites the memory area of the P-Box array				
1501h	Value range exceeded at LINE input				
1502h	Value range exceeded at ROW input				
1504h	Value range exceeded at POINT input				
1505h	Value range exceeded at LENGTH input				
1506h	Value range exceeded at MODE input				

**Example ST:**

```

(* Initialisation *)
if FirstTime then
  StringToPBox.ROW := 1;
  StringToPBox.Column := 16;
  FirstTime := False;
end_if;

(* Query actual position *)
ActPos(Enable := TRUE);
if ActPos.Valid then
  (* Display position in the PBox displays (PBox P1003 = PLC display ) *)
  DintToPBox.Value := ActPos.Position;
  DintToPBox.Column := 9;
  DintToPBox.LENGTH := 10;
  DintToPBox(Enable := True);
end_if;

(* Switch device on or off via DIG1 *)
Power(Enable := _5_State_digital_input.0);
if OldState <> Power.Status then
  OldState := Power.Status;
  (* Is device switched on? *)
  if Power.Status then
    StringToPBox(Enable := False, Text := TextOn);
  else
    StringToPBox(Enable := False, Text := TextOff);
  end_if;

  StringToPBox(Enable := TRUE);
else
  StringToPBox;
end_if;

```

**10.3.6.3 FB\_STRINGTOBOX**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

This function module copies a string (chain of characters) into the memory array of the ParameterBox. Via ROW and COLUMN the starting point of the string is set in the ParameterBox display. The parameter TEXT transfers the required string to the function module; the name of the string can be obtained from the table of variables. As long as ENABLE is set to 1, all changes to the inputs are adopted immediately. If the CLEAR input is set, the entire display content is overwritten with space characters before the selected string is written. If VALID changes to 1, the string has been correctly transferred. In case of error ERROR is set to 1. VALID is 0 in this case. In the ERRORID the relevant error code is then valid. With a negative flank on ENABLE, VALID, ERROR and ERRORID are reset.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Transfer of the string	BOOL	<b>VALID</b>	String transferred	BOOL
<b>CLEAR</b>	Clear display	BOOL	<b>ERROR</b>	Error in FB	BOOL
<b>ROW</b>	Line of the display Value range = 0 to 3	BYTE	<b>ERRORID</b>	Error code	INT
<b>COLUMN</b>	Column of the display Value range = 0 to 19	BYTE			
<b>TEXT</b>	Text to be displayed	STRING			
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1500h	String overwrites the memory area of the P-Box array				
1501h	Value range exceeded at ROW input				
1502h	Value range exceeded at COLUMN input				
1503h	The selected string number does not exist				
1506h	The option "Allow ParameterBox function modules" is not activated in the PLC configuration.				

**Example ST:**

```

(* Initialisation *)
if FirstTime then
  StringToPBox.ROW := 1;
  StringToPBox.Column := 16;
  FirstTime := False;
end_if;

(* Query actual position *)
ActPos(Enable := TRUE);
if ActPos.Valid then
  (* Display position in the PBox displays (PBox P1003 = PLC display) *)
  DintToPBox.Value := ActPos.Position;
  DintToPBox.Column := 9;
  DintToPBox.LENGTH := 10;
  DintToPBox(Enable := True);
end_if;

(* Switch device on or off via DIG1 *)
Power(Enable := _5_State_digital_input.0);
if OldState <> Power.Status then
  OldState := Power.Status;
  (* Is device switched on? *)
  if Power.Status then
    StringToPBox(Enable := False, Text := TextOn);
  else
    StringToPBox(Enable := False, Text := TextOff);
  end_if;

  StringToPBox(Enable := TRUE);
else
  StringToPBox;
end_if;

```

10.3.7 FB\_Capture (Detection of rapid events)

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X		

The cycle time of the PLC is 5 ms. This cycle may be too long to detect very rapid external events. Via FB Capture it is possible to capture certain physical values on flanks at the FI inputs. Monitoring of the inputs is carried out in a 1 ms cycle. The values which are saved can be read by the PLC later.

With a positive flank on EXECUTE all inputs are read in and the Capture function is enabled. The FI input which is to be monitored is selected via the INPUT input. Via EDGE, the type of flank and the behaviour of the module are selected.

**EDGE = 0** With the first positive flank, the selected value is saved under OUTPUT1 and DONE1 is set to 1. The next positive flank saves under OUTPUT2 and DONE2 is set to 1. The FB is then disabled.

**EDGE = 1** Behaviour as for EDGE = 0, with the difference that triggering is with the negative flank.

**EDGE = 2** With the first positive flank, the selected value is saved under OUTPUT1 and DONE1 is set to 1. The next positive flank saves under OUTPUT2 and DONE2 is set to 1. The FB is then disabled.

**EDGE = 3** Behaviour as for EDGE = 2, with the difference that triggering is first with the negative and then with the positive flank.

If the input CONTINUOUS is set to 1, then only the settings = and 1 are relevant to EDGE. The FB continues to run and always saves the last triggering event under OUTPUT1. DONE1 remains active as of the first event. DONE2 and OUTPUT2 are not used.

The BUSY output remains active until both Capture events (DONE1 and DONE2) have occurred.

The function of the module can be terminated at any time with a negative flank on EXECUTE. All outputs retain their values. With a positive flank on EXECUTE first, all outputs are deleted and then the function of the module is started.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>EXECUTE</b>	Execute	BOOL	<b>DONE1</b>	Value in OUTPUT1 valid	BOOL
<b>CONTINUOUS</b>	Single execution or continuous operation	BOOL	<b>DONE2</b>	Value in OUT valid	BOOL
<b>INPUT</b>	<b>SK54xE</b> Input to be monitored 0 = Input 1 ---- 7 = Input 8  <b>SK52xE, SK53xE, SK2xxE, SK2xxE FDS</b> 0 = Input 1 ---- 3 = Input 4	BYTE	<b>BUSY</b>	FB still waiting for a Capture event	BOOL
<b>EDGE</b>	Triggering flank	BYTE	<b>ERROR</b>	the FB has an error	BOOL
<b>SOURCE</b>	Value to be saved 0 = Position in rotations 1 = Actual frequency 2 = Torque	BYTE	<b>ERRORID</b>	Error code	INT
			<b>OUTPUT1</b>	Value for 1st Capture event	DINT
			<b>OUTPUT2</b>	Value for 2nd Capture event	DINT
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1900h	INPUT value range exceeded				
1901h	EDGE value range exceeded				
1902h	SOURCE value range exceeded				
1903h	More than two instances are active				

**Example ST:**

```

Power(ENABLE := TRUE);
IF Power.STATUS THEN
  Move(EXECUTE := TRUE, POSITION := Pos, VELOCITY := 16#2000);
  (* The FB waits for a High signal on DIG1. If this
     is detected, the FB saves the actual position. The value can
     be queried with the property "OUTPUT1". *)
  Capture(EXECUTE := TRUE, INPUT := 0);

  IF Capture.DONE1 THEN
    Pos := Capture.OUTPUT1;
    Move(EXECUTE := FALSE);
  END_IF;
END_IF;

```

**i Information**

Several instances of this FB may exist in the PLC program. However, only two instances may be active at the same time!

**10.3.8 FB\_DinCounter**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	V2.3 and above	V3.1 and above	V2.1 and above	X	V1.1 and above	

This function block is used to count pulses from digital inputs. All flanks (Low – High and High – Low) are counted. The minimum pulse width is 250µs.

The function block is activated with ENABLE. With the positive flank the inputs PV, UD, DIN and MODE are adopted and all outputs are deleted.

**UD** defines the counting direction

- 0 = larger numbers
- 1 = smaller numbers

A counter value can be entered in PV. This has different effects, depending on the setting for the MODE input.

**MODE**

- 0 = Overflow, the counter is used as a continuous counter. It can overflow in a positive or negative direction. When the function starts, CV = PV is set. In this mode, BUSY always remains at 1 and Q always remains 0.
- 1 = Without overflow
  - Forward counting: CV starts at 0, BUSY = 1, and runs until CV=>PV. BUSY then goes to 0 and Q to 1. The counting process stops.
  - Backward counting: CV starts at PV and runs until CV<=0. During this time BUSY = 1 and switches to 0 when the end of counting is reached. As a result, Q goes to 1.
  - The counter is restarted with a new flank at the ENABLE input

**DIN** defines the measurement input. The number of inputs depends on the particular FI.

- Input 1 = 0
- Input 2 = 1

etc.



VAR_INPUT			VAR_OUTPUT		
Input	Explanation	Type	Output	Explanation	Type
<b>ENABLE</b>	Enable	BOOL	<b>Q</b>	Counting complete	BOOL
<b>UD</b>	Counting direction  0 = larger numbers 1 = smaller numbers	BOOL	<b>BUSY</b>	Counter running	BOOL
<b>PV</b>	Counter value	INT	<b>ERROR</b>	the FB has an error	BOOL
<b>MODE</b>	Mode	BYTE	<b>ERRORID</b>	Error code	INT
<b>DIN</b>	Measurement input	BYTE	<b>CV</b>	Counter value	INT
			<b>CF</b>	Counting frequency (resolution 0.1)	INT
<b>ERRORID</b>	<b>Explanation</b>				
0	No error				
0x1E00	Digital input is already being used by another counter				
0x1E01	Digital input does not exist				
0x1E02	Number value range MODE exceeded				

### 10.3.9 FB\_FunctionCurve

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	

This function module produces a mapping control. Defined points can be communicated to the function block, with which it emulates a function. The output then behaves according to the saved map. Linear interpolation is carried out between the individual base points. The base points are defined with X and Y values. The X values are always of the INT type, the Y values can either be of the INT or the DINT type, depending on the size of the largest base point. More memory is required if DINT is used. The base points are entered in the column "Init Value" in the variables window. If TRUE is detected at the ENABLE input, on the basis of the input value INVALUE the corresponding output value OUTVALUE is calculated. VALID = TRUE indicates that the output value OUTVALUE is valid. As long as VALID is FALSE, the output OUTVALUE has the value 0. If the input value INVALUE exceeds the upper or the lower end of the characteristic range, the first or the last output value of the characteristic range remain until the INVALUE returns to within the area of the characteristic range. If the characteristic range is exceeded or undershot, the appropriate output MINLIMIT or MAXLIMIT is set to TRUE. ERROR becomes TRUE, if the abscissa values (X values) of the characteristic range do not continuously increase or if no table is initialised. The appropriate error is output by ERRORID and the starting value is 0. The error is reset if ENABLE = FALSE.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Execute	BOOL	<b>VALID</b>	Output value is valid	BOOL
<b>INVALUE</b>	Input value ( x )	INT	<b>ERROR</b>	Error in FB	BOOL
			<b>ERRORID</b>	Error code	INT
			<b>MAXLIMIT</b>	Max limit reached	BOOL
			<b>MINLIMIT</b>	Min limit reached	BOOL
			<b>OUTVALUE</b>	Output value ( y )	DINT
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1400h	Abscissa values (X values) of the characteristic range do not always increase				
1401h	No map initialised				

### 10.3.10 FB\_PIDT1

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

The P-I-DT1 is a freely parameterisable individual controller. If individual components or the P, I or DT1 component are not required, their parameters are written as 0. The T1 component only functions together with the D component. Therefore a PT1 controller cannot be parameterised. Due to internal memory limitations, the control parameters are restricted to the following areas:

Permissible value range for control parameters			
Parameter	Value range	Scaling	Resulting value range
<b>P (Kp)</b>	0 – 32767	1/100	0.00 – 32.767
<b>I (Ki)</b>	0 – 10240	1/100	0.00 – 10.240
<b>D (Kd)</b>	0 – 32767	1/1000	0.000 – 3.2767
<b>T1 (ms)</b>	0 – 32767	1/1000	0.000 – 3.2767
<b>Max</b>	-32768 – 32767		
<b>Min</b>	-32768 – 32767		

The controller starts to calculate when ENABLE is set to TRUE. The control parameters are only adopted with a rising flank from ENABLE. While ENABLE is TRUE, changes to the control parameters have no effect.. If ENABLE is set to FALSE, the output remains at its last value.

The output bit VALID is set, as long as the output value of Q is within the Min and Max limits and the ENABLE input is TRUE.

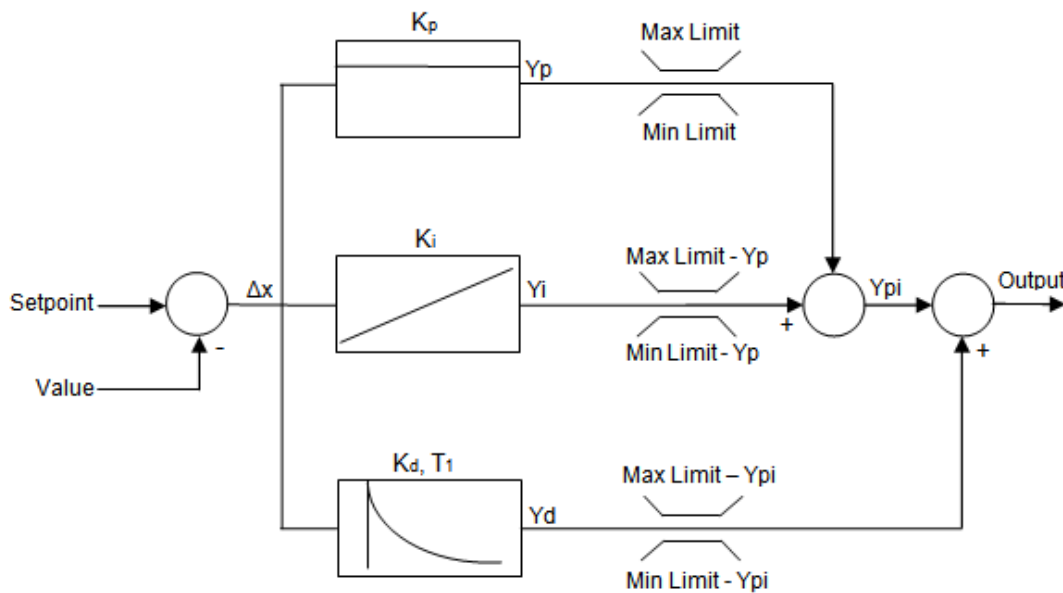
ERROR is set as soon as an error occurs. The VALID bit is then FALSE and the cause of the fault can be identified from the ERRORID (see table below).

If the RESET bit is set to TRUE, the content of the integrator and the differentiator are set to 0. If the ENABLE input is FALSE, the OUTPUT output is also set to 0. If the ENABLE input is set to TRUE, only the P component has an effect on the OUTPUT output.

If the output value OUTPUT is outside of the range of the maximum or minimum output values, the corresponding bit MAXLIMIT or MINLIMIT is set and the VALID bit is set to FALSE.

### **i** Information

If the entire program cannot be executed within a PLC cycle, the controller calculates the output value a second time with the old scanning values. This ensures a constant scanning rate. Because of this it is essential that the CAL command for the PIDT1 controller is executed in each PLC cycle and only at the end of the PLC program.



VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
<b>ENABLE</b>	Execute	BOOL	<b>VALID</b>	Output value is valid	BOOL
<b>RESET</b>	Reset outputs	BOOL	<b>ERROR</b>	Error in FB	BOOL
<b>P</b>	P component (Kp)	INT	<b>ERRORID</b>	Error code	INT
<b>I</b>	I component (Ki)	INT	<b>MAXLIMIT</b>	Maximum limit reached	BOOL
<b>D</b>	D component (Kd)	INT	<b>MINLIMIT</b>	Minimum limit reached	BOOL
<b>T1</b>	T1 component in ms	INT	<b>OUTPUT</b>	Output value	INT
<b>MAX</b>	Maximum output value	INT			
<b>MIN</b>	Minimum output value	INT			
<b>SETPOINT</b>	Setpoint	INT			
<b>VALUE</b>	Actual value	INT			
<b>ERRORID</b>	<b>Description</b>				
0	No error				
1600h	P component not within value range				
1601h	I component not within value range				
1602h	D component not within value range				
1603h	T1 component not within value range				

### 10.3.11 FB\_ResetPosition

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	V2.3 and above	V3.1 and above	V2.1 and above	X	V1.2 and above	

With a flank at the **EXECUTE** input, the actual position is set to the value which is entered in Position. With absolute encoders, the actual position can only be reset to 0. The value in Position is not used.

VAR_INPUT			VAR_OUTPUT		
Input	Explanation	Type	Output	Explanation	Type
<b>EXECUTE</b>	Execute	BOOL			
<b>Position</b>	Position	DINT			

### 10.3.12 FB\_Weigh

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	V2.3 and above	V3.1 and above	V2.1 and above	X	V1.2 and above	

This module is used to determine the average torque during movement at a constant speed. From this value, physical values, such as the weight which is being moved can be determined.

The FB is started via a positive flank on the EXECUTE input. With the flank, all inputs are adopted by the FB. The FI moves with the speed which is set in SPEED. The measurement is started after the elapse of the time which is set in STARTTIME. The duration of the measurement is defined under MEASURETIME. The FI stops after the elapse of the measurement time. If the input REVERSE = 1, the measurement process starts again, but with a negative speed. Otherwise the measurement is complete, the output DONE changes to 1 and the measurement result is in VALUE.

As long as the measurement process is running, BUSY is active. The scaling of the measurement result VALUE is 1 = 0.01% of the rated torque of the motor. Call-up of another Motion FB stops the measurement function and the output ABORT changes to 1. All outputs of the FB are reset with a new positive flank on EXECUTE.

VAR_INPUT			VAR_OUTPUT		
Input	Description	Type	Output	Description	Type
EXECUTE	Execute	BOOL	DONE	Measurement ended	BOOL
REVERSE	Change of rotation direction	BOOL	BUSY	Measurement running	BOOL
STARTTIME	Time to start of measurement in ms	INT	ABORT	Measurement aborted	BOOL
MEASURETIME	Measurement time in ms	INT	ERROR	the FB has an error	BOOL
SPEED	Measuring speed in % (standardised to the maximum frequency, 16#4000 corresponds to 100%)	INT	ERRORID	Error code	INT
			VALUE	Measurement result	INT
<b>ERRORID</b>	<b>Description</b>				
0	No error				
0x1000	FI not switched on				
0x1101	Setpoint frequency not parameterised as a setpoint (P553)				
0x1C00	STARTTIME value range exceeded				
0x1C01	MEASURETIME value range exceeded				
0x1C02	The tolerance of the measurement values with respect to each other is greater than 1/8				

**Example ST:**

```
(* Enable device *)
Power(Enable := TRUE);
(* Is the device enabled? *)
if Power.Status then
  (* Specify starting time 2000 ms *)
  Weigh.STARTTIME := 2000;
  (* Specify measuring time 1000 ms *)
  Weigh.MEASURETIME := 1000;
  (* Specify speed 25% of maximum speed *)
  Weigh.SPEED := 16#1000;
end_if;

Weigh(EXECUTE := Power.Status);
(* Was weighing completed? *)
if Weigh.done then
  Value := Weigh.Value;
end_if;
```

**i Information**

Only one instance of this FB is permissible in the PLC program!

**10.4 Operators**

**10.4.1 Arithmetical operators**

**i Information**

Some of the following operators may also contain further commands. These must be placed in brackets behind the operator. It must be noted that a space must be included behind the opened bracket. The closing bracket must be placed on a separate line of the program.

```
LD Var1
ADD( Var2
SUB Var3
)
```

**10.4.1.1 ABS**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>			X	X		

Forms the absolute value from the accumulator.

**Example AWL:**

```
LD -10 (* Load the value -10 *)
ABS (* Accumulator = 10 *)
ST Value1 (* Saves the value 10 in Value1 *)
```

**Example ST:**

```
Value1 := ABS(-10); (* The result is 10 *)
```

### 10.4.1.2 ADD and ADD(

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>		X	X	X		

Adds variables and constants together with the correct prefixes. The first value for addition is in the AE/accumulator, the second is loaded with the ADD command or is inside the bracket. Several variables or constants can be added to the ADD command. For bracket addition, the accumulator is added to the result of the expression in brackets. Up to 6 bracket levels are possible. The values to be added must belong to the same type of variable.

#### Example AWL:

```
LD 10
ADD 204          (* Addition of two constants *)
ST Value
LD 170          (* Addition of a constant and 2 variables. *)
ADD Var1, Var2 (* 170dez + Var1 + Var2 *)
ST Value
LD Var1
ADD( Var2
SUB Var3        (* Var1 + ( Var2 - Var3 ) *)
)
ST Value
```

#### Example ST:

```
Result := 10 + 30; (* The result is 40 *)
Result := 10 + Var1 + Var2;
```

### 10.4.1.3 DIV and DIV(

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>		X	X	X		

Divides the accumulator by the operands For divisions by zero, the maximum possible result is entered into the accumulator, e.g. for a division with INT values, this is the value 0x7FFF or the value 0x8000 if the divisor is negative. For bracket division, the accumulator is divided by the result of the expression in brackets. Up to 6 bracket levels are possible. The values to be divided must belong to the same type of variable.

#### Example AWL:

```
LD 10
DIV 3          (* Division of two constants *)
ST iValue
LD 170        (* The result is 9 *)
LD 170        (* Division of a constant and 2 variables. *)
DIV Var1, Var2 (* (170dez : Var1) : Var2 *)
ST Value
LD Var1
DIV( Var2
SUB Var3
)              (* Divide Var1 by the content of the brackets *)
ST Value
)              (* Var1 : ( Var2 - Var3 ) *)
```

**Example ST:**

```
Result := 30 / 10; (* The result is 3 *)
Result := 30 / Var1 / Var2;
```

**10.4.1.4 LIMIT**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type		X	X	X		

The command limits the value in the accumulator to the transferred minimum and maximum values. If this is exceeded, the maximum value is entered in the accumulator and if it is undershot, the minimum value is entered. If the value lies between the limits, there is no effect.

**Example AWL:**

```
LD 10 (* Loads the value 10 into the accumulator *)
LIMIT 20, 30 (* The value is compared with the limits 20 and 30. *)
(* The value in the accumulator is smaller, the Accumulator is overwritten
with 20 *)
ST iValue (* Saves the value 20 in Value1 *)
```

**Example ST:**

```
Result := Limit(10, 20, 30); (* The result is 20 *)
```

**10.4.1.5 MAX**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type		X	X	X		

This value determines the maximum value of two variables or constants. For this, the current value of the accumulator is compared with the value transferred in the MAX command. After the command, the larger of the two values is in the accumulator. Both values must belong to the same type of variable.

**Example AWL:**

```
LD 100 (* Load 100 into the accumulator *)
MAX 200 (* Compare with the value 200 *)
ST iValue (* Save 200 in Value2 (because larger value) *)
```

**Example in ST:**

```
Result := Max(100, 200); (* The result is 200 *)
```

**10.4.1.6 MIN**

	SK 54xE	SK 53xE	SK 2xxE	SK 2xxE-FDS	SK 180E	SK 155E-FDS
--	---------	---------	---------	-------------	---------	-------------



		SK 52xE				SK 190E	SK 175E-FDS
Availability	X	X	X	X	X	X	
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>			
Data type		X	X	X			

This command determines the minimum value of two variables or constants. For this, the current value of the accumulator is compared with the value transferred in the MIN command. After the command, the smaller of the two values is in the accumulator. Both values must belong to the same type of variable.

**Example AWL:**

```
LD 100      (* Load 100 into the accumulator *)
MIN 200     (* Compare with the value 200 *)
ST Value2  (* Save 100 in Value2 (because smaller value) *)
```

**Example ST::**

```
Result := Min(100, 200); (* Save 100 in Value2 (because smaller value) *)
```

**10.4.1.7 MOD and MOD(**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
Data type		X	X	X		

The Accumulator is divided by one or more variables or constant and the remainder of the division is the result in the accumulator. For the bracket Modulus, the accumulator is divided by the result of the expression in the brackets and the modulus is formed from this. Up to 6 bracket levels are possible.

**Example AWL:**

```
LD 25      (* Load the dividend *)
MOD 20     (* Division 25/20 to Modulus = 5 *)

ST Var1   (* Save result 5 in Var1 *)

LD 25      (* Load the dividend *)
MOD( Var1 (* Result = 25/(Var1 + 10) to Modulus in the Accu *)
ADD 10
)
ST Var3   (* Save result 10 in Var3 *)
```

**Example ST:**

```
Result := 25 MOD 20;          (* Save result 5 in Var1 *)
Result := 25 MOD (Var1 + 10); (* Result = 25/(Var1 + 10) per modulus into the Accumulator *)
```

**10.4.1.8 MUL and MUL(**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Data type		X	X	X

Multiplication of the accumulator with one or more variables or constants. For bracket multiplication, the accumulator is multiplied by the result of the expression in brackets. Up to 6 bracket levels are possible. Both values must belong to the same type of variable.

**Example AWL:**

```
LD 25          (* Load the multiplier *)
MUL Var1, Var2 (* 25 * Var1 * Var2 *)
ST Var2        (* Save result *)

LD 25          (* Load the multiplier *)
MUL( Var1      (* Result = 25*(Var1 + Var2) *)
ADD Var2
)
ST Var3        (* Save result as variable Var3 *)
```

**Example ST:**

```
Result := 25 * Var1 * Var2;
Result := 25 * (Var1 + Var2);
```

**10.4.1.9 MUX**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Data type		X	X	X

Various constants or variables can be selected via an index, which is located in front of the command in the accumulator. The first value is accessed via the Index 0. The selected value is loaded into the accumulator. The number of values is only limited by the program memory.

**Example AWL:**

```
LD 1          (* Select the required element *)
MUX 10,20,30,40,Value1 (* MUX command with 4 constants and a variable *)
ST Value      (* Save result = 20 *)
```

**Example ST:**

```
Result := Mux(1, 10, 20, 30, 40, Value1) (* Save result = 20 *)
```

**10.4.1.10 SUB and SUB(**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Data type		X	X	X

Subtracts the accumulator from one or more variables or constants. For bracket subtraction, the accumulator is subtracted from the result of the expression in brackets. Up to 6 bracket levels are possible. The values to be subtracted must belong to the same type of variable.

**Example AWL:**

```
LD 10
SUB Var1          (* Result = 10 - Var1 *)
ST Result


LD 20
SUB Var1, Var2, 30 (* Result = 20 - Var1 - Var2 - 30 *)
ST Result

LD 20
SUB( 6            (* Subtract 20 from the contents of the bracket *)
AND 2
)
ST Result          (* Result = 20 - (6 AND 2) *)
```

**Example ST:**

```
Result := 10 - Value1;
```

**10.4.2 Extended mathematical operators**

** Information**

The operators listed here require intensive computing. This may result in a considerably longer running time for the PLC program.

**10.4.2.1 COS, ACOS, SIN, ASIN, TAN, ATAN**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X		
<b>Data type</b>	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
				X		

Calculation of the relevant mathematical function. The value to be calculated must be available in minutes of arc. The scaling corresponds to 1 = 1000.

Conversion: Angle in radians = (Angle in degrees \* PI / 180)\*1000  
 e.g. an angle of 90° is converted as follows: 90° \* 3.14 / 180) \*1000 = 1571

$$AE = \sin\left(\frac{AE}{1000}\right) \cdot 1000 \quad AE = \cos\left(\frac{AE}{1000}\right) \cdot 1000 \quad AE = \tan\left(\frac{AE}{1000}\right) \cdot 1000$$

**Example AWL:**

```
LD 1234
SIN
ST Result (* Result = 943 *)
```

**Example ST:**

```

Result := COS(1234); (* Result = 330 *)
Result := ACOS(330); (* Result = 1234 *)
Result := SIN(1234); (* Result = 943 *)
Result := ASIN(943); (* Result = 1231 *)
Result := TAN(999); (* Result = 1553 *)
Result := ATAN(1553); (* Result = 998 *)

```

10.4.2.2 EXP

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X		
	BOOL	BYTE	INT	DINT		
Data type				X		

Forms the exponential function to the base of Euler's Number (2.718) from the Accumulator. Up to 3 places behind the decimal point may be stated, i.e. 1.002 must be entered as 1002.

$$AE = e^{\left(\frac{AE}{1000}\right)} \cdot 1000$$

Example AWL:

```

LD 1000
EXP
ST Result (* Result = 2718 *)

```

Example ST:

```

Result := EXP(1000); (* Result = 2718 *)

```

10.4.2.3 LN

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X		
	BOOL	BYTE	INT	DINT		
Data type				X		

Logarithm to base e (2.718). Up to 3 places behind the decimal point may be stated, i.e. 1.000 must be entered as 1000.

$$AE = \ln\left(\frac{AE}{1000}\right) \cdot 1000$$

Example AWL:

```

LD 1234
LN
ST Result

```

**Example ST:**

```
Result := LN(1234); (* Result = 210 *)
```

**10.4.2.4 LOG**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X		
	BOOL	BYTE	INT	DINT		
Data type				X		

Forms the base 10 logarithm from the accumulator. Up to 3 places behind the decimal point may be stated, i.e. 1.000 must be entered as 1000.

$$AE = \log_{10} \left( \frac{AE}{1000} \right) \cdot 1000$$

**Example AWL:**

```
LD 1234
LOG
ST Result (* Result = 91 *)
```

**Example ST:**

```
Result := LOG(1234); (* Result = 91 *)
```

**10.4.2.5 SQRT**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X		
	BOOL	BYTE	INT	DINT		
Data type				X		

Forms the square root from the accumulator. Up to 3 places behind the decimal point may be stated, i.e. 1.000 must be entered as 1000.

$$AE = \sqrt{\left( \frac{AE}{1000} \right) \cdot 1000}$$

**Example AWL:**

```
LD 1234
SQRT
ST Result (* Result = 1110 *)
```

**Example ST:**

```
Result := SQRT(1234); (* Result = 1110 *)
```

### 10.4.3 Bit operators

#### 10.4.3.1 AND and AND()

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
Data type	X	X	X	X		

Bit-wise AND link of the AE/accumulator with one or two variables or constants. Bit-wise AND(...) linking with the AE/accumulator and the AE/accumulator which was previously formed in the bracket. Up to 6 bracket levels are possible. All values must belong to the same type of variable.

#### Example in AWL:

```
LD 170
AND 204 (* AND link between 2 constants *)
(* Akku = 136 (see the example below the table) *)

LD 170 (* link between a constant and 2 variables.*)
AND Var1, Var2 (* Akku = 170dez AND Var1 AND Var2 *)

LD Var1
AND ( Var2 (* AE/Akku = Var1 AND ( Var2 OR Var3 ) *)
OR Var3
)
```

#### Example in ST:

```
Result := 170 AND 204; (* Result = 136dec *)
```

Var2	Var1	Result
0	0	0
0	1	0
1	0	0
1	1	1

Example: 170dez (1010 1010bin) AND 204dez (1100 1100bin) = (1000 1000bin) 136dez

#### 10.4.3.2 ANDN and ANDN()

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
Data type	X	X	X	X		

Bit-wise AND linking of the AE/accumulator with a negated operand. Bit-wise AND (...) linking of the AE/accumulator and the negated result of the bracket. Up to 6 bracket levels are possible. The values to be linked must belong to the same type of variable.

**Example AWL:**

```
LD 2#0000_1111
ANDN 2#0011_1010 (* ANDN link between 2 constants *)
(* Accu = 2#1111_0101 *)

LD 170 (* Link between a constant and 2 variables. *)
ANDN Var1, Var2 (* Accu = 170d ANDN Var1 ANDN Var2 *)

LD Var1
ANDN ( Var2 (* AE/Accu = Var1 ANDN ( Var2 OR Var3 ) *)
OR Var3
)
```

Var2	Var1	Result
0	0	1
0	1	1
1	0	1
1	1	0

**10.4.3.3 NOT**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>	X	X	X	X		

Bit-wise negation of the accumulator.

**Example AWL:**

```
LD BYTE#10 (* Load the value 10dec into the ACCU in Byte format *)
NOT (* The value is resolved on the Bit level (0000 1010), *)
(* Negated bit-wise (1111 0101) and then converted back *)
(* to a decimal value, result = 245dec *)
ST Var3 (* Save result as variable Var3 *)
```

**Example ST:**

```
Result := not BYTE#10; (* Result = 245dec *)
```

**10.4.3.4 OR and OR(**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		

Data type	X	X	X	X
-----------	---	---	---	---

Bit-wise OR link of the AE/accumulator with one or two variables or constants. Bit-wise OR(...) linking with the AE/accumulator and the AE/accumulator which was previously formed in the bracket. Up to 6 bracket levels are possible. All values must belong to the same type of variable.

**Example AWL:**

```
LD 170
OR 204      (* OR link between 2 constants *)

LD 170      (* Link between a constant and 2 variables. *)
OR Var1, Var2 (* Accu = 170d OR Var1OR Var2 *)

LD Var1
OR ( Var2   (* AE/Accu = Var1 OR ( Var2 AND Var3 ) *)
AND Var3
)
```

**Example ST:**

```
Result := 170 or 204; (* Result = 238 *)
```

Var2	Var1	Result
0	0	0
0	1	1
1	0	1
1	1	1

**10.4.3.5 ORN and ORN(**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Data type	X	X	X	X

Bit-wise OR linking of the AE/accumulator with a negated operand. Bit-wise OR (...) linking of the AE/accumulator and the negated result of the bracket. Up to 6 bracket levels are possible. The values to be linked must belong to the same type of variable.

**Example AWL:**

```
LD 2#0000_1111
ORN 2#0011_1010 (* ORN link between 2 constants *)
(* Accu = 2#1100_0000 *)

LD 170      (* Link between a constant and 2 variables. *)
ORN Var1, Var2 (* Accu = 170d ORN Var1 ORN Var2 *)

LD Var1
ORN ( Var2   (* AE/Accu = Var1 ORN ( Var2 OR Var3 ) *)
OR Var3
)
```



**Example ST:**

```
Result := 2#0000_1111 ORN 2#0011_1010; (* Result = 2#1100_0000 *)
```

Var2	Var1	Result
0	0	1
0	1	0
1	0	0
1	1	0

**10.4.3.6 ROL**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>		X	X	X		

Bit-wise rotation of the accumulator to the left The content of the accumulator is shifted n times to the left, whereby the left bit is inserted again on the right.

**Example AWL:**

```
LD 175 (* Loads the value 1010_1111*)
ROL 2 (* Accu content is rotated 2x to the left *)
ST Value1 (* Saves the value 1011_1110 *)
```

**Example ST:**

```
Result := ROL(BYTE#175, 2); (* Result = 2#1011_1110 *)
Result := ROL(INT#175, 2); (* Result = 16#C02B *)
```

**10.4.3.7 ROR**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>		X	X	X		

Bit-wise rotation of the accumulator to the right. The content of the accumulator is shifted n times to the right, whereby the right bit is inserted again on the left.

**Example AWL:**

```
LD 175 (* Loads the value 1010_1111 *)
ROR 2 (* Accu content is rotated 2x to the right *)
ST Value1 (* Saves the value 1110_1011 *)
```

**Example ST:**

```
Result := ROR(BYTE#175, 2); (* Result = 2#1110_1011 *)
```

**10.4.3.8 S and R**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type	X					

Sets and resets a boolean variable if the result of the previous link (the AE) was TRUE.

**Example AWL:**

```
LD TRUE (* Loads the AE with TRUE *)
S Var1 (* VAR1 is set to TRUE *)
R Var1 (* VAR1 is set to FALSE *)
```

**Example ST:**

```
Result := TRUE;
Result := FALSE;
```

**10.4.3.9 SHL**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type		X	X	X		

Bit-wise left shift of the accumulator. The content of the accumulator is shifted n times to the left and the bits which are pushed out are lost.

**Example AWL:**

```
LD 175 (* Loads the value 1010_1111 *)
SHL 2 (* Accu content is shifted 2x to the left *)
ST Value1 (* Saves the value 1011_1100 *)
```

**Example ST:**

```
Result := SHL(BYTE#175, 2); (* Result = 2#1011_1100 *)
Result := SHL(INT#175, 2); (* Result = 16#2BC *)
```

**10.4.3.10 SHR**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type		X	X	X		

Bit-wise right shift of the accumulator. The content of the accumulator is shifted n times to the right and the bits which are pushed out are lost.

**Example AWL:**

```
LD 175      (* Loads the value 1010_1111 *)
SHR 2      (* Accu content is shifted 2 x to the right *)
ST Value1 (* Saves the value 0010_1011 *)
```

**Example ST:**

```
Result := SHR(BYTE#175, 2); (* Result = 2#0010_1011 *)
```

**10.4.3.11 XOR and XOR(**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>	X					

Bit-wise "exclusive OR" link between the AE/accumulator and one or two variables or constants. The first value is located in the AE/accumulator and the second is loaded with the command or is within the brackets. The values to be linked must belong to the same type of variable.

**Example AWL:**

```
LD 2#0000_1111
XOR 2#0011_1010 (* XOR link between 2 constants *)
(* Accu = 2#0011_0101 *)

LD 170          (* Link between a constant and 2 variables. *)
XOR Var1, Var2 (* Accu = 170d XOR Var1 XOR Var2 *)

LD Var1
XOR ( Var2      (* AE/Accu = Var1 XOR ( Var2 OR Var3 ) *)
OR Var3
)
```

**Example ST:**

```
Result := 2#0000_1111 XOR 2#0011_1010; (* Result = 2#0011_0101 *)
```

Var2	Var1	Result
0	0	0
0	1	1
1	0	1
1	1	0

**10.4.3.12 XORN and XORN(**

	SK 54xE	SK 53xE	SK 2xxE	SK 2xxE-FDS	SK 180E	SK 155E-FDS
--	---------	---------	---------	-------------	---------	-------------

		SK 52xE				SK 190E	SK 175E-FDS
Availability	X	X	X	X	X	X	
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>			
Data type	X						

Bit-wise Exclusive OR linking of the AE/accumulator with a negated operand. Bit-wise Exclusive OR (...) linking of the AE/accumulator and the negated result of the bracket. Up to 6 bracket levels are possible. The values to be linked must belong to the same type of variable.

**Example AWL:**

```
LD 2#0000_1111
XORN 2#0011_1010 (* XORN link between 2 constants *)
(* Accu = 2#1100_1010 *)

LD 170 (* Link between a constant and 2 variables. *)
XORN Var1, Var2 (* Accu = 170d XORN Var1 XORN Var2 *)

LD Var1
XORN ( Var2 (* AE/Accu = Var1 XORN ( Var2 OR Var3 ) *)
OR Var3
)
```

**Example ST:**

```
Result := 2#0000_1111 XORN 2#0011_1010; (* Result = 2#1100_1010 *)
```

Var2	Var1	Result
0	0	1
0	1	0
1	0	0
1	1	1

**10.4.4 Loading and storage operators (AWL)**

**10.4.4.1 LD**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
Data type	X	X	X	X		

Loads a constant or a variable into the AE or the accumulator.

**Example AWL:**

```
LD 10 (* Loads 10 as BYTE *)
LD -1000 (* Loads -1000 as INTEGER *)
LD Value1 (* Loads variable Value 1 *)
```

### 10.4.4.2 LDN

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type	X	X	X	X		

Loads a negated boolean variable into the AE.

#### Example AWL:

```
LDN Value1 (* Value1 = TRUE à AE = FALSE *)
ST Value2 (* Speicher auf Value2 = FALSE *)
```

### 10.4.4.3 ST

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type	X	X	X	X		

Stores the content of the AE/accumulator to a variable. The variable to be saved must match the previously loaded and processed data type.

#### Example AWL:

```
LD 100 (* Loads the value 1010_1111 *)
ST Value1 (* Accumulator content 100 is saved in Value1 *)
```

### 10.4.4.4 STN

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type	X	X	X	X		

Stores the content of the AE to a variable and negates it. The variable to be saved must match the previously loaded and processed data type.

#### Example AWL:

```
LD Value1 (* Value1 = TRUE à AE = TRUE *)
STN Value2 (* Store to Value2 = FALSE *)
```

## 10.4.5 Comparison operators

10.4.5.1 EQ

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type		X	X	X		

Compares the content of the accumulator with a variable or constant. If the values are equal, the AE is set to TRUE.

**Example AWL:**

```
LD Value1 (* Value1 = 5 *)
EQ 10 (* AE = Is 5 equal to 10 ? *)
JMPC NextStep (* AE = FALSE - program does not jump *)
ADD 1
NextStep:
ST Value1
```

**Example ST:**

```
(* Ist Value = 10 *)
if Value = 10 then
  Value2 := 5;
end_if;
```

10.4.5.2 GE

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type		X	X	X		

Compares the content of the accumulator with a variable or constant. If the value in the accumulator is greater or equal to the variable or constant, then AE is set to TRUE.

**Example AWL:**

```
LD Value1 (* Value1 = 5 *)
GE 10 (* Is 5 greater than or equal to 10? *)
JMPC NextStep (* AE = FALSE - program does not jump *)
ADD 1

NextStep:
ST Value1
```

**Example ST:**

```
(* Is 5 greater than or equal to 10? *)
if Value >= 10 then
  Value := Value - 1
end_if;
```

10.4.5.3 GT

	SK 54xE	SK 53xE	SK 2xxE	SK 2xxE-FDS	SK 180E	SK 155E-FDS
--	---------	---------	---------	-------------	---------	-------------

		SK 52xE				SK 190E	SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X	
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>			
<b>Data type</b>		X	X	X			

Compares the content of the accumulator with a variable or constant. If the value in the accumulator is greater than the variable or constant, the AE is set to TRUE.

**Example AWL:**

```
LD Value1(* Value1 = 12 *)
GT 8 (* Is 12 greater than 8? *)
JMPC NextStep (* AE = TRUE - program jumps *)
ADD 1
NextStep:
ST Value1
```

**Example ST:**

```
(* Is 12 greater than 8? *)
if Value > 8 then
  Value := 0;
end_if;
```

**10.4.5.4 LE**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>		X	X	X		

Compares the content of the accumulator with a variable or constant. If the value in the Accumulator is less than or equal to the variable or constant, then AE is set to TRUE.

**Example AWL:**

```
LD Value1 (* Value1 = 5 *)
LE 10 (* Is 5 less than or equal to 10? *)
JMPC NextStep:
ST Value1
```

**Example ST:**

```
(* Is 5 less than or equal to 10? *)
if Value <= 10 then
  Value := 11;
end_if;
```

**10.4.5.5 LT**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		

Data type		X	X	X
-----------	--	---	---	---

Compares the content of the accumulator with a variable or constant. If the value in the accumulator is less than the variable or constant the AE is set to TRUE.

**Example AWL:**

```
LD Value1 (* Value1 = 12 *)
LT 8 (* Is 12 less than 8? *)
JMPC NextStep (* AE = FALSE - program does not jump *)
ADD 1
NextStep:
ST Value1
```

**Example ST:**

```
(* Is 12 less than 8? *)
if Value < 0 then
  Value := 0;
end_if;
```

10.4.5.6 NE

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Data type		X	X	X

Compares the content of the accumulator with a variable or constant. If the value in the Accumulator is not equal to the variable or constant, the AE is set to TRUE.

**Example AWL:**

```
LD Value1 (* Value1 = 5 *)
NE 10 (*Is 5 not equal to 10?*)
JMPC NextStep (* AE = TRUE - program jumps *)
ADD 1
NextStep:
ST Value1
```

**Example ST:**

```
if Value <> 5 then
  Value := 5;
end_if;
```

10.5 Processing values

All analogue and digital inputs and outputs or bus setpoints and actual values can be read and processed by the PLC or can be set by the PLC (if they are output values). Access to the individual values is via the process values listed below. For all output values, the output (e.g. digital outputs or PLC setpoint) must be programmed so that the PLC is the source of the event. All process data is read in from the PLC by the device at the start of each cycle and is only written to the device at the end of the program. The following table lists all of the values which can be directly accessed by the PLC. All other process values must be accessed via the function blocks MC\_ReadParameter or MC\_WriteParameter.



### 10.5.1 Inputs and outputs

Here, all process values which describe the I/O interface of the device are listed.

Name	Function	Standardisation	Type	Access	Device
_0_Set_digital_output	Set digital outputs	Bit 0: Mfr1 Bit 1: Mfr2 Bit 2: Dout1 Bit 3: Dout2 Bit 4: dig. Fct. Aout Bit 5: Dout3 (Din7) Bit 6: Statusword Bit 8 Bit 7: Statusword Bit 9 Bit 8: BusIO Bit0 Bit 9: BusIO Bit1 Bit 10: BusIO Bit2 Bit 11: BusIO Bit3 Bit 12: BusIO Bit4 Bit 13: BusIO Bit5 Bit 14: BusIO Bit6 Bit 15: BusIO Bit7	UINT	R/W	SK 54xE
_0_Set_digital_output	Set digital outputs	Bit 0: Mfr 1 Bit 1: Mfr 2 Bit 2: DOUT1 Bit 3: DOUT2 Bit 4: Dig. Analog Out Bit 5: free Bit 6: Bus PZD Bit 10 Bit 7: Bus PZD Bit 13 Bit 8: BusIO Bit0 Bit 9: BusIO Bit1 Bit 10: BusIO Bit2 Bit 11: BusIO Bit3 Bit 12: BusIO Bit4 Bit 13: BusIO Bit5 Bit 14: BusIO Bit6 Bit 15: BusIO Bit7	UINT	R/W	SK 52xE SK 53xE
_0_Set_digital_output	Set digital outputs	Bit 0: DOUT1 Bit 1: BusIO Bit0 Bit 2: BusIO Bit1 Bit 3: BusIO Bit2 Bit 4: BusIO Bit3 Bit 5: BusIO Bit4 Bit 6: BusIO Bit5 Bit 7: BusIO Bit6 Bit 8: BusIO Bit7 Bit 9: Bus PZD Bit 10 Bit 10: Bus PZD Bit 13 Bit 11: DOUT2	UINT	R/W	SK 2xxE
_0_Set_digital_output	Set digital outputs	Bit 0: DOUT1 Bit 1: DOUT2	UINT	R/W	SK 180E SK 190E

Name	Function	Standardisation	Type	Access	Device
		Bit 2: BusIO Bit0 Bit 3: BusIO Bit1 Bit 4: BusIO Bit2 Bit 5: BusIO Bit3 Bit 6: BusIO Bit4 Bit 7: BusIO Bit5 Bit 8: BusIO Bit6 Bit 9: BusIO Bit7 Bit 10: Bus PZD Bit 10 Bit 11: Bus PZD Bit 13			SK 2xxE-FDS
_0_Set_digital_output	Set digital outputs	Bit 0: DOUT1 Bit 1: DOUT2 Bit 2: DOUT_BRAKE Bit 3: DOUT_BUS1 Bit 4: DOUT_BUS2	UINT	R/W	SK 155E-FDS SK 175E-FDS
_1_Set_analog_output	Set analogue output 1. IOE	10.0V = 100	BYTE	R/W	SK 54xE SK 53xE SK 52xE
_2_Set_external_analog_out1	Set analogue output 1. IOE	10.0V = 100	BYTE	R/W	SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_3_Set_external_analog_out2	Set analogue output 2. IOE	10.0V = 100	BYTE	R/W	SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_4_State_digital_output	State of digital outputs	Bit 0: Mfr1 Bit 1: Mfr2 Bit 2: Dout1 Bit 3: Dout2 Bit 4: dig. Fkt. Aout Bit 5: Dout3 (Din7) Bit 6: Bus Status word Bit 8 Bit 7: Status word Bit 9 Bit 8: BusIO Bit0 Bit 9: BusIO Bit1 Bit 10: BusIO Bit2 Bit 11: BusIO Bit3 Bit 12: BusIO Bit4 Bit 13: BusIO Bit5 Bit 14: BusIO Bit6 Bit 15: BusIO Bit7	INT	R	SK 54xE
_4_State_digital_output	State of digital outputs	P711	BYTE	R	SK 52xE SK 53xE SK 2xxE

Name	Function	Standardisation	Type	Access	Device
					SK 2xxE-FDS SK 180E SK 190E
_4_State_digital_output	State of digital outputs	Bit 0: DOUT1 Bit 1: DOUT2 Bit 2: DOUT_BRAKE Bit 3: DOUT_BUS1 Bit 4: DOUT_BUS2	BYTE	R	SK 155E-FDS SK 175E-FDS
_5_State_Digital_input	State of digital inputs	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: DIN4 Bit 4: DIN5 Bit 5: DIN6 Bit 6: DIN7 Bit 7: Digital function AIN1 Bit 8: Digital function AIN2	INT	R	SK 54xE
_5_State_Digital_input	State of digital inputs	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: DIN4 Bit 4: DIN5 Bit 5: DIN6 Bit 6: DIN7	INT	R	SK 52xE SK 53xE
_5_State_Digital_input	State of digital inputs	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: DIN4/AIN1 Bit 4: AIN2 Bit 5: PTC Bit 6: free Bit 7: free Bit 8: DIN1 IOE 1 Bit 9: DIN2 IOE 1 Bit 10: DIN3 IOE 1 Bit 11: DIN4 IOE 1 Bit 12: DIN1 IOE 2 Bit 13: DIN2 IOE 2 Bit 14: DIN3 IOE 2 Bit 15: DIN4 IOE 2	INT	R	SK 2xxE SK 180E SK 190E
_5_State_Digital_input	State of digital inputs	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: TF (PTC) Bit 4: DIN-BUS1 (ASi1) Bit 5: DIN-BUS2 (ASi2) Bit 6: DIN-BUS3 (ASi3)	INT	R	SK 155E-FDS SK 175E-FDS

Name	Function	Standardisation	Type	Access	Device
		Bit 7: DIN-BUS4 (ASi4) Bit 8: STO Bit 9: BDDI1 (ASIO3) Bit10: BDDI2 (ASIO4)			
_5_State_Digital_input	State of digital inputs	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: DIN4 Bit 4: DIN5 Bit 5: DIN6/AIN1 Bit 6: DIN7/AIN2 Bit 7: PTC Bit 8: DIN1 IOE 1 Bit 9: DIN2 IOE 1 Bit 10: DIN3 IOE 1 Bit 11: DIN4 IOE 1 Bit 12: DIN1 IOE 2 Bit 13: DIN2 IOE 2 Bit 14: DIN3 IOE 2 Bit 15: DIN4 IOE 2	INT	R	SK 2xxE-FDS
_6_Delay_digital_inputs	State of digital inputs according to P475	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: DIN4 Bit 4: DIN5 Bit 5: DIN6 Bit 6: DIN7 Bit 7: Digital function AIN1 Bit 8: Digital function AIN2	INT	R	SK 54xE
_6_Delay_digital_inputs	State of digital inputs according to P475	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: DIN4 Bit 4: DIN5 Bit 5: DIN6 Bit 6: DIN7	INT	R	SK 52xE SK 53xE
_6_Delay_digital_inputs	State of digital inputs according to P475	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: AIN1 Bit 4: AIN2 Bit 5: PTC Bit 6: free Bit 7: free Bit 8: DIN1 IOE 1 Bit 9: DIN2 IOE 1 Bit 10: DIN3 IOE 1 Bit 11: DIN4 IOE 1	INT	R	SK 2xxE SK 180E SK 190E

Name	Function	Standardisation	Type	Access	Device
		Bit 12: DIN1 IOE 2 Bit 13: DIN2 IOE 2 Bit 14: DIN3 IOE 2 Bit 15: DIN4 IOE 2			
_6_Delay_digital_inputs	State of digital inputs according to P475	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: DIN4 Bit 4: DIN5 Bit 5: DIN6/AIN1 Bit 6: DIN7/AIN2 Bit 7: PTC Bit 8: DIN1 IOE 1 Bit 9: DIN2 IOE 1 Bit 10: DIN3 IOE 1 Bit 11: DIN4 IOE 1 Bit 12: DIN1 IOE 2 Bit 13: DIN2 IOE 2 Bit 14: DIN3 IOE 2 Bit 15: DIN4 IOE 2	INT	R	SK 2xxE-FDS
_7_Analog_input1	Value of analogue input 1 (AIN1)	10.00V = 1000	INT	R	all
_8_Analog_input2	Value of analogue input 2 (AIN2)	10.00V = 1000	INT	R	all
_9_Analog_input3	Value of analogue function DIN2	10.00V = 1000	INT	R	SK 54xE SK 155E-FDS SK 175E-FDS
_10_Analog_input4	Value of analogue function DIN3	10.00V = 1000	INT	R	SK 54xE SK 155E-FDS SK 175E-FDS
_11_External_analog_input1	Value of analogue input 1 (1.IOE)	10.00V = 1000	INT	R	SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_12_External_analog_input2	Value of analogue input 2 (1.IOE)	10.00V = 1000	INT	R	SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_13_External_analog_input3	Value of analogue input 1 (2.IOE)	10.00V = 1000	INT	R	SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_14_External_analog_input4	Value of analogue input 2 (2.IOE)	10.00V = 1000	INT	R	SK 54xE SK 2xxE

Name	Function	Standardisation	Type	Access	Device
					SK 2xxE-FDS SK 180E SK 190E
_15_State_analog_output	Status of analogue output	10.0V = 100	BYTE	R	SK 54xE
_16_State_ext_analog_out1	Status of analogue output (1. IOE)	10.00V = 1000	INT	R	SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_17_State_ext_analog_out2	Status of analogue output (2. IOE)	10.00V = 1000	INT	R	SK 54xE SK 2xxE SK 180E SK 190E
_18_Dip_switch_state	Status of DIP switch	Bit 0: DIP1 Bit 1: DIP2 Bit 2: DIP3 Bit 3: DIP4 Bit 4: DIP_I1 Bit 5: DIP_I2 Bit 6: DIP_I3 Bit 7: DIP_I4	INT	R	SK 155E-FDS SK 175E-FDS

### 10.5.2 PLC setpoints and actual values

The process values listed here form the interface between the PLC and the device. The function of the PLC setpoints is specified in (P553).

#### **i** Information

The process value PLC\_control\_word overwrites the function block MC\_Power. The PLC setpoints overwrite the function blocks MC\_Move.... and MC\_Home.

Name	Function	Standardisation	Type	Access	Device
_20_PLC_control_word	PLC Control word	Corresponds to USS profile	INT	R/W	all
_21_PLC_set_val1	PLC setpoint 1	100% = 4000h	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_22_PLC_set_val2	PLC setpoint 2	100% = 4000h	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE

Name	Function	Standardisation	Type	Access	Device
					SK 2xxE-FDS SK 180E SK 190E
_23_PLC_set_val3	PLC setpoint 3	100% = 4000h	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_24_PLC_set_val4	PLC setpoint 4	100% = 4000h	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_25_PLC_set_val5	PLC setpoint 5	100% = 4000h	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_26_PLC_additional_control_word1	PLC additional control word 1	Corresponds to USS profile	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_27_PLC_additional_control_word2	PLC additional control word 2	Corresponds to USS profile	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_28_PLC_status_word	PLC status word	Corresponds to USS profile	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_29_PLC_act_val1	PLC actual value 1	100% = 4000h	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E

Name	Function	Standardisation	Type	Access	Device
_30_PLC_act_val2	PLC actual value 2	100% = 4000h	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_31_PLC_act_val3	PLC actual value 3	100% = 4000h	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_32_PLC_act_val4	PLC actual value 4	100% = 4000h	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_33_PLC_act_val5	PLC actual value 5	100% = 4000h	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_34_PLC_Busmaster_Control_word	Master function control word (bus master function) via PLC	Corresponds to USS profile	INT	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_35_PLC_32Bit_set_val1	32Bit PLC setpoint - P553[1] = Low Part of 32Bit value - P553[2] = High Part of 32Bit value	–	LONG	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_36_PLC_32Bit_act_val1	32Bit PLC actual value - PLC actual value 1 = Low part of 32Bit value - PLC actual value 2 = High part of 32Bit value	–	LONG	R/W	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_37_PLC_status_bits	Virtual PLC status outputs	Bit 0: PLC-DOUT1 Bit 1: PLC-DOUT2	INT	R/W	SK 155E-FDS SK 175E-FDS
_38_PLC_control_bits	Virtual PLC control	Bit 0: PLC-DIN1	INT	R/W	SK 155E-FDS



Name	Function	Standardisation	Type	Access	Device
	outputs	Bit 1: PLC-DIN2 Bit 2: PLC-DIN3 Bit 3: PLC-DIN4 Bit 4: PLC-DIN5 Bit 5: PLC-DIN6 Bit 6: PLC-DIN7 Bit 7: PLC-DIN8			SK 175E-FDS

### 10.5.3 Bus setpoints and actual values

These process values reflect all setpoints and actual values which are transferred to the device via the various bus systems.

Name	Function	Standardisation	Type	Access	Device
_40_Inverter_status	FI status word	Corresponds to USS profile	INT	R	all
_41_Inverter_act_val1	FI actual value 1	100% = 4000h	INT	R	all
_42_Inverter_act_val2	FI actual value 2	100% = 4000h	INT	R	all
_43_Inverter_act_val3	FI actual value 3	100% = 4000h	INT	R	all
_44_Inverter_act_val4	FI actual value 4	100% = 4000h	INT	R	SK 54xE
_45_Inverter_act_val5	FI actual value 5	100% = 4000h	INT	R	SK 54xE
_46_Inverter_lead_val1	Broadcast Master Function: Master value 1	100% = 4000h	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_47_Inverter_lead_val2	Broadcast Master Function: Master value 2	100% = 4000h	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_48_Inverter_lead_val3	Broadcast Master Function: Master value 3	100% = 4000h	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_49_Inverter_lead_val4	Broadcast Master Function: Master value 4	100% = 4000h	INT	R	SK 54xE
_50_Inverter_lead_val5	Broadcast Master	100% = 4000h	INT	R	SK 54xE

Name	Function	Standardisation	Type	Access	Device
	Function: Master value 5				
_51_Inverter_control_word	Resulting bus control word	Corresponds to USS profile	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_52_Inverter_set_val1	Resulting main bus setpoint 1	100% = 4000h	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_53_Inverter_set_val2	Resulting main bus setpoint 2	100% = 4000h	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_54_Inverter_set_val3	Resulting main bus setpoint 3	100% = 4000h	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_55_Inverter_set_val4	Resulting main bus setpoint 4	100% = 4000h	INT	R	SK 54xE
_56_Inverter_set_val5	Resulting main bus setpoint 5	100% = 4000h	INT	R	SK 54xE
_57_Broadcast_set_val1	Broadcast Slave: Auxiliary setpoint 1	100% = 4000h	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_58_Broadcast_set_val2	Broadcast Slave: Auxiliary setpoint 2	100% = 4000h	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_59_Broadcast_set_val3	Broadcast Slave: Auxiliary setpoint 3	100% = 4000h	INT	R	SK 54xE SK 53xE

Name	Function	Standardisation	Type	Access	Device
					SK 52xE SK 2xxE SK 180E SK 190E
_60_Broadcast_set_val4	Broadcast Slave: Auxiliary setpoint 4	100% = 4000h	INT	R	SK 54xE
_61_Broadcast_set_val5	Broadcast Slave: Auxiliary setpoint 5	100% = 4000h	INT	R	SK 54xE
_62_Inverter_32Bit_set_val1	Resulting 32Bit main setpoint 1 Bus	- Low part in P546[1] - High part in P546[2]	LONG	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_63_Inverter_32Bit_act_val1	FI 32Bit actual value 1	- Low part in P543[1] - High part in P543[2]	LONG	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_64_Inverter_32Bit_lead_val1	32Bit lead value 1	- Low part in P502[1] - High part in P502[2]	LONG	R	SK 54xE SK 2xxE SK 180E SK 190E
_65_Broadcast_32Bit_set_val1	32Bit Broadcast Slave auxiliary setpoint 1	- Low part in P543[1] - High part in P543[2]	LONG	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_66_BusIO_input_bits	Incoming Bus I/O data	- Bit 0 – 7 = Bus I/O In Bit 0 – 7 - Bit 8 = Flag 1 - Bit 9 = Flag 2 - Bit 10 = Bit 8 of Bus control word - Bit 11 = Bit 9 of Bus control word	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E

#### 10.5.4 ControlBox and ParameterBox

The control boxes can be accessed via the process values listed here. This enables the implementation of simple HMI applications.

Name	Function	Standardisation	Type	Access	Device
_70_Set_controlbox_show_val	Display value for the ControlBox	Display value = Bit 29 – Bit 0	DINT	R/W	all

		Decimal point position = Bit 31 - Bit 30			
_71_Controlbox_key_state	ControlBox keyboard status	Bit 0: ON Bit 1: OFF Bit 2: DIR Bit 3: UP Bit 4: DOWN Bit 5: Enter	BYTE	R	all
_72_Parameterbox_key_state	ParameterBox keyboard status	Bit 0: ON Bit 1: OFF Bit 2: DIR Bit 3: UP Bit 4: DOWN Bit 5: Enter Bit 6: Right Bit 7: Left	BYTE	R	all

### 10.5.5 Info parameters

The most important actual values for the device are listed here.

Name	Function	Standardisation	Type	Access	Device
_80_Current_fault	Number of actual fault	Error 10.0 = 100	BYTE	R	All
_81_Current_warning	Actual warning	Warning 10.0 = 100	BYTE	R	All
_82_Current_reason_FI_blocked	Actual reason for switch-on block state	Problem 10.0 = 100	BYTE	R	All
_83_Input_voltage	Actual mains voltage	100 V = 100	INT	R	All
_84_Current_frequenz	Actual frequency	10Hz = 100	INT	R	All
_85_Current_set_point_frequency1	Actual setpoint frequency from the setpoint source	10Hz = 100	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_86_Current_set_point_frequency2	Actual inverter setpoint frequency	10Hz = 100	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_87_Current_set_point_frequency3	Actual setpoint frequency after ramp	10Hz = 100	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E

Name	Function	Standardisation	Type	Access	Device
					SK 190E
_88_Current_Speed	Calculated actual speed	100rpm = 100	INT	R	All
_89_Actual_current	Actual output current	10.0A = 100	INT	R	All
_90_Actual_torque_current	Actual torque current	10.0A = 100	INT	R	All
_91_Current_voltage	Actual voltage	100V = 100	INT	R	All
_92_Dc_link_voltage	Actual link circuit voltage	100V = 100	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_93_Actual_field_current	Actual field current	10.0A = 100	INT	R	All
_94_Voltage_d	Actual voltage component d-axis	100V = 100	INT	R	All
_95_Voltage_q	Actual voltage component q-axis	100V = 100	INT	R	All
_96_Current_cos_phi	Actual Cos(phi)	0.80 = 80	BYTE	R	all
_97_Torque	Actual torque	100% = 100	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_98_Field	Actual field	100% = 100	BYTE	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_99_Apparent_power	Actual apparent power	1.00KW = 100	INT	R	All
_100_Mechanical_power	Actual mechanical power	1.00KW = 100	INT	R	All
_101_Speed_encoder	Actual measured speed	100rpm = 100	INT	R	SK 54xE SK 53xE SK 52xE
_102_Usage_rate_motor	Actual motor usage rate (instantaneous value)	100% = 100	INT	R	all
_103_Usage_rate_motor	Actual motor usage rate	100% = 100	INT	R	SK 54xE

Name	Function	Standardisation	Type	Access	Device
r_I2t	I2t				SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_104_Usage_rate_brake_resistor	Actual brake resistor usage rate	100% = 100	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_105_Head_sink_temp	Actual heat sink temperature	100°C = 100	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_106_Inside_temp	Actual inside temperature	100°C = 100	INT	R	SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_107_Motor_temp	Actual motor temperature	100°C = 100	INT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_108_Actual_net_frequency	Actual net frequency	10Hz = 100	INT	R	SK 155E-FDS SK 175E-FDS
_109_Mains_phase_sequence	Mains phase sequence	0=CW, 1=CCW	BYTE	R	SK 155E-FDS SK 175E-FDS
_141_Pos_Sensor_Inc	Position of incremental encoder	0.001 rotation	DINT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_142_Pos_Sensor_Abs	Position of absolute encoder	0.001 rotation	DINT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_143_Pos_Sensor_Uni	Position of universal encoder	0.001 rotation	DINT	R	SK 54xE

Name	Function	Standardisation	Type	Access	Device
_144_Pos_Sensor_HTL	Position of HTL encoder	0.001 rotation	DINT	R	SK 54xE
_145_Actual_pos	Actual position	0.001 rotation	DINT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_146_Actual_ref_pos	Actual setpoint position	0.001 rotation	DINT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_147_Actual_pos_diff	Difference in position between setpoint and actual value	0.001 rotation	DINT	R	SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E

### 10.5.6 PLC errors

The device errors E23.0 to E23.2 can be set from the PLC program via the User Error Flags.

Name	Function	Standardisation	Type	Access	Device
_110_ErrorFlags	Generates user error in device	Bit 0: E 23.0 Bit 1: E 23.1 Bit 2: E 23.2 Bit 3: E 23.3 Bit 4: E 23.4 Bit 5: E 23.5 Bit 6: E 23.6 Bit 7: E 23.7	BYTE	R/W	all
_111_ErrorFlags_ext	Generates user error in device	Bit 0: E 24.0 Bit 1: E 24.1 Bit 2: E 24.2 Bit 3: E 24.3 Bit 4: E 24.4 Bit 5: E 24.5 Bit 6: E 24.6 Bit 7: E 24.7	BYTE	R/W	all

### 10.5.7 PLC parameters

The PLC parameters P355, P356 and P360 can be directly accessed via this group of process data.

Name	Function	Standardisation	Type	Access	Device
_115_PLC_P355_1	PLC INT parameter P355 [-01]	-	INT	R	all
_116_PLC_P355_2	PLC INT parameter P355 [-02]	-	INT	R	all
_117_PLC_P355_3	PLC INT parameter P355 [-03]	-	INT	R	all
_118_PLC_P355_4	PLC INT parameter P355 [-04]	-	INT	R	all
_119_PLC_P355_5	PLC INT parameter P355 [-05]	-	INT	R	all
_120_PLC_P355_6	PLC INT parameter P355 [-06]	-	INT	R	all
_121_PLC_P355_7	PLC INT parameter P355 [-07]	-	INT	R	all
_122_PLC_P355_8	PLC INT parameter P355 [-08]	-	INT	R	all
_123_PLC_P355_9	PLC INT parameter P355 [-09]	-	INT	R	all
_124_PLC_P355_10	PLC INT parameter P355 [-10]	-	INT	R	all
_125_PLC_P356_1	PLC LONG parameter P356 [-01]	-	DINT	R	all
_126_PLC_P356_2	PLC LONG parameter P356 [-02]	-	DINT	R	all
_127_PLC_P356_3	PLC LONG parameter P356 [-03]	-	DINT	R	all
_128_PLC_P356_4	PLC LONG parameter P356 [-04]	-	DINT	R	all
_129_PLC_P356_5	PLC LONG parameter P356 [-05]	-	DINT	R	all
_130_PLC_P360_1	PLC display parameter P360[-01]	-	DINT	R/W	all
_131_PLC_P360_2	PLC display parameter P360[-02]	-	DINT	R/W	all
_132_PLC_P360_3	PLC display parameter P360[-03]	-	DINT	R/W	all
_133_PLC_P360_4	PLC display parameter P360[-04]	-	DINT	R/W	all
_134_PLC_P360_5	PLC display parameter P360[-05]	-	DINT	R/W	all
_135_PLC_Scope_Int_1	PLC scope display value 1	-	INT	R/W	all
_136_PLC_Scope_Int_1	PLC scope display value	-	INT	R/W	all



Name	Function	Standardisation	Type	Access	Device
2	2				
_137_PLC_Scope_Int_3	PLC scope display value 3	-	INT	R/W	all
_138_PLC_Scope_Int_4	PLC scope display value 4	-	INT	R/W	all
_139_PLC_Scope_Bool_1	PLC scope display value 5	-	INT	R/W	all
_140_PLC_Scope_Bool_2	PLC scope display value 6	-	INT	R/W	all

## 10.6 Languages

### 10.6.1 Instruction list (AWL / IL)

#### 10.6.1.1 General

##### Data types

The PLC supports the data types listed below.

Name	Required memory space	Value range
BOOL	1 Bit	0 to 1
BYTE	1 Byte	0 to 255
INT	2 Byte	-32768 to 32767
DINT	4 Byte	-2147483648 to 2147483647
LABEL_ADDRESSES	2 Byte	Jump marks

##### Literal

For greater clarity it is possible to enter constants for all data types in various display formats. The following table gives an overview of all possible variants.

Literal	Example	Number displayed in decimal
<b>Bool</b>	FALSE	0
	TRUE	1
	BOOL#0	0
	BOOL#1	1
<b>Dual (Base 2)</b>	2#01011111	95
	2#0011_0011	51
	BYTE#2#00001111	15
	BYTE#2#0001_1111	31
<b>Oktal (Base 8)</b>	8#0571	377
	8#05_71	377
	BYTE#8#10	8
	BYTE#8#111	73
	BYTE#8#1_11	73
<b>Hexadecimal (Base 16)</b>	16#FFFF	-1
	16#0001_FFFF	131071
	INT#16#1000	4096
	DINT#16#0010_2030	1056816
<b>Integer (Base 10)</b>	10	10
	-10	-10
	10_000	10000
	INT#12	12
	DINT#-100000	-100000
<b>Time</b>	TIME#10s50ms	10.050 seconds
	T#5s500ms	5.5 seconds
	TIME#5.2s	5.2 seconds
	TIME#5D10H15M	5days+10hours+15minutes
	T#1D2H30M20S	1day+2hours+30minutes+20seconds

### Comments

It is advisable to provide the sections of the program with comments in order to make the PLC program understandable at a later date. In the application program these comments are marked by starting with the character sequence "(" and finishing with ")" as shown in the following examples.

### Example:

```
(* Comment about a program block *)
LD 100 (* Comment after a command *)
ADD 20
```

### Jump marks

With the aid of the operators JMP, JMPC or JMPCN whole sections of the program can be bypassed. A jump mark is given as the target address. With the exception of diacritics and „ß“ it may contain all letters, the numbers 0 to 9 and underscores; other characters are not permitted. The jump mark is terminated with a colon. This may stand on its own. There may also be further commands after in the same line after the jump mark.

Possible variants may appear as follows:

#### Example:

```
Jump mark:
LD 20

This_is_a_jumpmark:
ADD 10

MainLoop: LD 1000
```

A further variant is the transfer of a jump mark as a variable. This variable must be defined as type LABEL\_ADDRESS in the variable table, then this can be loaded into the variable 'jump marks'. With this, status machines can be created very simply, see below.

#### Example:

```
LD FirstTime
JMPC AfterFirstTime
(* The label address must be initialized at the beginning *)
LD Address_1
ST Address_Var
LD TRUE
ST FirstTime
AfterFirstTime:

JMP Address_Var

Address_1:
LD Address_2
ST Address_Var
JMP Ende

Address_2:
LD Address_3
ST Address_Var
JMP Ende

Address_3:
LD Address_1
ST Address_Var

Ende:
```

### Function call-ups

The Editor supports one form of function call-ups. In the following version, the function CTD is called up via the instance I\_CTD. The results are saved in variables. The meaning of the functions used below is described in further detail later in the manual.

#### Example

```
LD 10000
ST I_CTD.PV
LD LoadNewVar
ST I_CTD.LD
LD TRUE
ST I_CTD.CD
CAL I_CTD
LD I_CTD.Q
ST ResultVar
LD I_CTD.CV
ST CurrentCountVar
```

**Bit-wise access to variables**

A simplified form is possible for access to a bit from a variable or a process variable.

Command	Description
LD Var1.0	Loads Bit 0 of Var1 into the AE
ST Var1.7	Stores the AE on Bit 7 of Var1
EQ Var1.4	Compares the AE with Bit 4 of Var1

**10.6.2 Structured text (ST)**

Structured text consists of a series of instruction, which are executed as in plain language ("IF..THEN..ELSE) or in loops (WHILE.. DO).

**Example:**

```
IF value < 7 THEN
  WHILE value < 8 DO
    value := value + 1;
  END_WHILE;
END_IF;
```

**10.6.2.1 Common**

**Data types (ST)**

The PLC supports the data types listed below.

Name	Required memory space	Value range
BOOL	1 Bit	0 to 1
BYTE	1 Byte	0 to 255
INT	2 Byte	-32768 to 32767
DINT	4 Byte	-2147483648 to 2147483647

### Assignment operator

On the left hand side of an assignment there is an operand (variable, address) to which the value of an expression on the right hand side is assigned with the assignment operator "=".

#### Example:

```
Var1 := Var2 * 10;
```

After execution of this line, Var1 has ten times the value of Var2.

### Call-up of function blocks in ST

A function block is called in ST by writing the name of the instance of the function block and then assigning the values of the parameters in brackets. In the following example a timer is called up with assignment of its parameters IN and PT Then the result variable Q is assigned to the variable A.

The result variable is accessed as in IL with the name of the function block, a following period and the name of the variable.

#### Example:

```
Timer(IN := TRUE, PT := 300);  
A := Timer.Q;
```

### Evaluation of expressions

The evaluation of the expression is performed by processing the operators according to certain linking rules. The operator with the strongest link is processed first and then the operator with the next strongest link, etc. until all of the operators have been processed. Operators with links of the same strength are processed from left to right.

The table below shows the ST operators in the order of the strength of their links:

Operation	Symbol	Link strength
Brackets	(Expression)	Strongest
Function call	Function name (parameter list)	
Negated complement formation	NOT	
Multiply Divide Modulus AND	* / MOD AND	
Add Subtract OR XOR	+ - OR XOR	
Compare Equality Inequality	<,>,<=,>= = <>	Light

### 10.6.2.2 Procedure

#### Return

The RETURN instruction can be used to jump to the end of the program, for example, depending on a condition.

#### IF

With the IF instruction, a condition can be tested and instructions carried out depending on this condition.

#### Syntax:

```
IF <Boolean_Expression1> THEN
  <IF_Instruction>
ELSIF <Boolean_Expression2> THEN
  <ELSIF_Instruction1>
ELSIF <Boolean_Expression n> THEN
  <ELSIF_Instruction n-1>
ELSE
  <ELSE_Instruction>}
END_IF;
```

The part in the curly brackets {} is optional. If <Boolean\_Expression1> is TRUE, then only the <IF\_Instructions> are executed and none of the other instructions.. Otherwise, starting with <Boolean\_Expression2>, the boolean expressions are evaluated in sequence until one of the expressions is TRUE. Then, only the expressions following this boolean expression and before the next ELSE or ELSIF are evaluated. If none of the boolean expressions is TRUE, only the <ELSE\_Instructions> are evaluated.

#### Example:

```
IF temp < 17 THEN
  Bool1 := TRUE;
ELSE
  Bool2 := FALSE;
END_IF;
```

#### CASE

With the CASE instruction, several conditional instructions with the same condition variables can be combined into a construct.

#### Syntax:

```
CASE <Var1> OF
  <Value1>: <Instruction 1>
  <Value2>: <Instruction 2>
  <Value3, Value4, Value5>: <Instruction 3>
  <Value6 .. Value10 >: <Instruction 4>
  ...
  <Value n>: <Instruction n>
ELSE <ELSE-Instruction>
END_CASE;
```

A CASE instruction is processed according to the following pattern:

- If the variable in <Var1> has the value <Value i>, the instruction <Instruction i> is executed.
- If <Var 1> does not have any of the stated values, the <ELSE instruction> is executed.
- If the same instruction is to be executed for several values of the variable, these values can be written separately in sequence, separated with commas as the condition of the common instruction.
- If the same instruction is to be executed for a range of values of the variable, the initial value and the end value can be written separated by a colon as the condition for the common instruction.

### Example:

```

CASE INT1 OF
  1, 5:
    BOOL1 := TRUE;
    BOOL3 := FALSE;
  2:
    BOOL2 := FALSE;
    BOOL3 := TRUE;
  10..20:
    BOOL1 := TRUE;
    BOOL3:= TRUE;
  ELSE
    BOOL1 := NOT BOOL1;
    BOOL2 := BOOL1 OR BOOL2;
END_CASE;

```

### FOR loop

Repetitive processes can be programmed with the FOR loop.

#### Syntax:

```

FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <STEP>} DO
  <Instruction>
END_FOR;

```

The part in the curly brackets {} is optional. The <Instructions> are executed as long as the counter <INT-Var> is not larger than the <END\_VALUE>. This is checked before the execution of the <Instructions> so that the <Instructions> are never executed if the <INIT\_VALUE> is larger than the <END\_VALUE>. Whenever the <Instructions> are executed, the <INIT-Var> is increased by a <Step size>. The step size can have any integer value. If this is missing, it is set to 1. The loop must terminate, as <INT\_Var> is larger.

### Example:

```

FOR Zaehler :=1 TO 5 BY 1 DO
  Var1 := Var1 * 2;
END_FOR;

```

### REPEAT loop

The REPEAT loop is different from the WHILE loop in that the termination condition is only tested after the loop has been executed. As a result, the loop must be run through at least once, regardless of the termination condition.

#### Syntax:

```

REPEAT
  <Instruction>
UNTIL
  <Boolean Expression>
END_REPEAT;

```

The <Instructions> are executed until the <Boolean Expression> is TRUE. If the <Boolean Expression> is TRUE with the first evaluation, the <Instructions> are executed exactly once. If the <Boolean Expression> is never TRUE, the <Instructions> will be executed endlessly, which will create a runtime error.

**i Information**

The programmer must ensure that no endless loops are created by changing the condition in the instruction part of the loop, for example a counter which counts upwards or downwards.

**Example:**

```
REPEAT
  Var1 := Var1 * 2;
  Count := Count - 1;
UNTIL
  Count = 0
END_REPEAT
```

**WHILE loop**

The WHILE loop can be used in the same way as the FOR loop, with the difference that the termination condition can be any boolean expression. This means that a condition is stated, which, if it is true, will result in the execution of the loop.

**Syntax:**

```
WHILE <Boolean Expression> DO
  <Instructions>
END_WHILE;
```

The <Instructions> are executed repeatedly for as long as the <Boolean\_Expression> is TRUE. If the <Boolean\_Expression> is FALSE in the first evaluation, the <Instructions> will never be executed. If the <Boolean\_Expression> is never FALSE, the <Instructions> will be repeated endlessly.

**i Information**

The programmer must ensure that no endless loops are created by changing the condition in the instruction part of the loop, for example a counter which counts upwards or downwards.

**Example:**

```
WHILE Count <> 0 DO
  Var1 := Var1*2;
  Count := Count - 1;
END_WHILE
```

**Exit**

If the EXIT instruction occurs in a FOR, WHILE or REPEAT loop, the innermost loop will be terminated, regardless of the termination condition.

**10.7 Jumps**

**10.7.1 JMP**

	SK 54xE	SK 53xE	SK 2xxE	SK 2xxE-FDS	SK 180E	SK 155E-FDS
--	---------	---------	---------	-------------	---------	-------------



		SK 52xE			SK 190E	SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

Unconditional jump to a jump mark.

**Example AWL:**

```
JMP NextStep (* Unconditional jump to NextStep *)
ADD 1

NextStep:
ST Value1
```

**10.7.2 JMPC**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

Conditional jump to a jump point If AE = TRUE, the command JMPC jumps to the stated jump point.

**Example AWL:**

```
LD 10
JMPC NextStep (* AE = TRUE - program jumps *)
ADD 1

NextStep:
ST Value1
```

**10.7.3 JMPCN**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X

Conditional jump to a jump point JMPCN jumps if the AE register = FALSE. Otherwise the program continues with the next instruction.

**Example AWL:**

```
LD 10
JMPCN NextStep (* AE = TRUE - program does not jump *)
ADD 1

NextStep:
ST Value1
```

**10.8 Type conversion**

**10.8.1 BOOL\_TO\_BYTE**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS

<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>	X					

Converts the data type of the AE from BOOL to BYTE. If the AE is FALSE, the accumulator is converted to 0. If the AE is TRUE, the accumulator is converted to 1.

**Example AWL:**

```
LD TRUE
BOOL_TO_BYTE (* AE = 1 *)
```

**Example ST:**

```
Result := BOOL_TO_BYTE(TRUE); (* Result = 1 *)
```

### 10.8.2 BYTE\_TO\_BOOL

	<b>SK 54xE</b>	<b>SK 53xE SK 52xE</b>	<b>SK 2xxE</b>	<b>SK 2xxE-FDS</b>	<b>SK 180E SK 190E</b>	<b>SK 155E-FDS SK 175E-FDS</b>
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>		X				

Converts the data type from BYTE to BOOL. As long as BYTE is not equal to zero, this always gives the conversion result TRUE.

**Example AWL:**

```
LD 10
BYTE_TO_BOOL (* AE = TRUE *)
```

**Example ST:**

```
Result := BYTE_TO_BOOL(10); (* Result = TRUE *)
```

### 10.8.3 BYTE\_TO\_INT

	<b>SK 54xE</b>	<b>SK 53xE SK 52xE</b>	<b>SK 2xxE</b>	<b>SK 2xxE-FDS</b>	<b>SK 180E SK 190E</b>	<b>SK 155E-FDS SK 175E-FDS</b>
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>		X				

Converts the data type from BYTE to INT. The BYTE is copied into the Low component of the INT and the High component of INT is set to 0.

**Example AWL:**

```
LD 10
BYTE_TO_INT (* Akku = 10 *)
```

**Example ST:**

```
Result := BYTE_TO_INT(10); (* Result = 10 *)
```

**10.8.4 DINT\_TO\_INT**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>				X		

Converts the data type from DINT to INT. The High component of the DINT value is not transferred.

**Example AWL:**

```
LD 200000
DINT_TO_INT (* Akku = 3392 *)

LD DINT# -5000
DINT_TO_INT (* Akku = -5000 *)

LD DINT# -50010
DINT_TO_INT (* Akku = 15526 *)
```

**Example ST:**

```
Result := DINT_TO_INT(200000); (* Result = 3392 *)
Result := DINT_TO_INT(-5000); (* Result = -5000 *)
Result := DINT_TO_INT(-50010); (* Result = 15526 *)
```

**10.8.5 INT\_TO\_BYTE**

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Availability</b>	X	X	X	X	X	X
	<b>BOOL</b>	<b>BYTE</b>	<b>INT</b>	<b>DINT</b>		
<b>Data type</b>			X			

Converts the data type from INT to BYTE. Here, the High component of the INT value is not transferred. Prefixes are lost as the BYTE type does not have prefixes.

**Example AWL:**

```
LD 16#5008
INT_TO_BYTE (* Akku = 8 *)
```

**Example ST:**

```
Result := INT_TO_BYTE(16#5008); (* Result = 8 *)
```

### 10.8.6 INT\_TO\_DINT

	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Availability	X	X	X	X	X	X
	BOOL	BYTE	INT	DINT		
Data type			X			

Converts the data type from INT to DINT. The INT is copied into the Low component of the DINT and the High component of the DINT is set to 0.

#### Example AWL:

```
LD 10
INT_TO_DINT (* Akku = 10 *)
```

#### Example ST:

```
Result := INT_TO_DINT(10); (* Result = 10 *)
```

## 10.9 PLC Error messages

Error messages cause the device to switch off, in order to prevent a device fault. With PLC error messages execution by the PLC is stopped and the PLC goes into the status "PLC Error". With other error messages the PLC continues operation. The PLC restarts automatically after the error has been acknowledged.

### The PLC continues to operate with PLC User Fault 23.X!

SimpleBox		Fault	
Group	Details in P700[-01] / P701	Text in the ParameterBox	Cause Remedy
E022	22.0	No PLC program	The PLC has been started but there is no PLC program in the device - Load PLC program into the FI
	22.1	PLC program is faulty	The checksum check via the PLC program produced an error. - Restart the device (Power ON) and try again - Alternatively, reload PLC program
	22.2	Incorrect jump address	Program error, behaviour as for Error 22.1
	22.3	Stack overflow	More than 6 bracket levels were opened during the run time of the program - Check the program for run time errors
	22.4	Max. PLC cycles exceeded	The stated maximum cycle time for the PLC program was exceeded - Change the cycle time or check the program

SimpleBox		Fault	Cause Remedy
Group	Details in P700[-01] / P701	Text in the ParameterBox	
	22.5	Unknown command code	A command code in the program cannot be executed because it is not known. - Program error, behaviour as for Error 22.1 - Version of the PLC and the NORD CON version do not match
	22.6	PLC write access	The program content has been changed while the PLC program was running
	22.9	PLC Error	The cause of the fault cannot be precisely determined - Behaviour as in Error 22.1
E023	23.0	PLC User Fault 1	This error can be triggered by the PLC program in order to externally indicate problems in the execution of the PLC program. Triggered by writing the process variable "ErrorFlags".
	23.1	PLC User Fault 2	
	23.2	PLC User Fault 3	

## 11 Project Mode

### 11.1 General

The project mode is an extension of the normal mode. As default, this is deactivated and must be activated in the Settings. The mode allows the user to manage a project. Projects can be loaded and saved. A project includes the frequency inverters with their data (parameters and PLC program), links to external parameter files or PLC programs, as well as the layout of the application. With a new start of NORD CON the last project which has been saved is always loaded. A new project is created if no project can be found. The project mode was developed for the following application:

- 11.2 "HMI"
- 11.3 "Save and restore"

Category	Name	Description
File	New project	This action creates an empty project.
	Open project...	This action opens a file selection dialogue and the user must select a project file (*.ncpx).
	Save project	This action opens a file selection dialogue and the user specifies a name for the project file (*.ncpx). After this, the project is saved under this name.
	Save all	The action
Project	Send all data	This action sends all parameters and the PLC program to the devices.
	Read all data	The action loads all parameters from the devices and saves them in the project file. In addition, the PLC program in the device is compared with that in the project. If they are not identical a warning is displayed in the log.
	Remove parameter	This action deletes the parameter for the selected device from the project.
	Remove PLC program	The action deletes the PLC program for the selected device from the project.
	Add PLC program	This action adds a saved PLC program for the highlighted device.
	Export parameters	This action exports the parameters for the selected device to a file.
	Export PLC program	This action exports the PLC program for the selected device to a file.
	PLC	Save
Save as ...		This action opens a file selection dialogue and the user must select a file name. After this, the PLC program is saved in a separate file.
Parameter setup	Save	This action saves the parameters in the project file.
	Save as ...	This action opens a file selection dialogue and the user must select a file

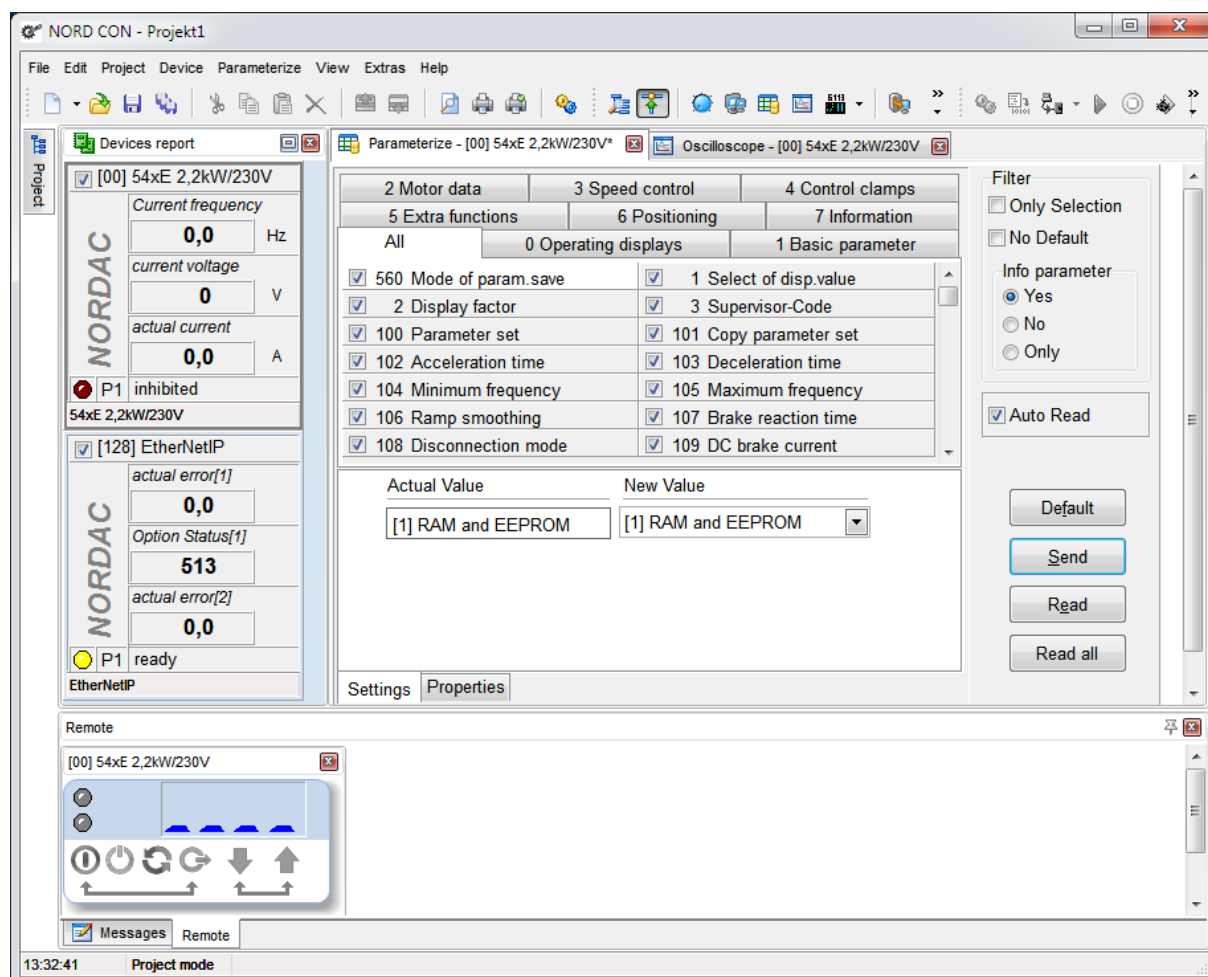
Category	Name	Description
		name. After this, the parameters are saved in a separate file.

## 11.2 HMI

The project mode is ideally suitable for good visualisation. The user connects the PC to the system and starts the device search (Bus scan Ctrl. F5). After the device search is complete, the user can place the required display elements for the device in the work area, e.g. parameter window, oscilloscope or control window. After this, the project must be saved. After the project has been opened, the devices and the layout are restored. This enables the user to always work with the same user interface.

### Information

When a project is loaded, it is not checked whether the devices which are contained in the project are actually connected. Communication errors may result if other devices are connected to the bus. Please take care that you always use the same device for the communication connection if you use the system bus.



## 11.3 Save and restore

The project mode can also be used to save and restore parameters and PLC programs. The list of devices used can be further restricted after a device search (bus scan). By deactivating the device in the device overview, a device can be excluded from saving and restoring.

The procedure may take several minutes, as depending on the particular system, the device list may contain several devices. The progress of the action is displayed in a separate window. NORD CON cannot be used during this process.

### Information

When a project is loaded, it is not checked whether the devices which are contained in the project are actually connected. Communication errors may result if other devices are connected to the bus. Please take care that you always use the same device for the communication connection if you use the system bus.

### Save

After a device search (bus scan) the action "Read all Data" reads the parameters of all of the devices which are found. The parameters must first be saved in NORD CON and then saved manually in the project file (Save All). Three options are available to users for the action "Read all Data". These options can be activated or deactivated in the settings dialogue.

Option	Description
Delete all data records on cancellation	If this option is activated, the data records for all of the devices which are included in the project are deleted when the function "Read all Data" is cancelled. Otherwise, not all parameters will be read out and the data in the project file is incomplete.
Delete incomplete data records	If this option is enabled, the data record for a device is deleted if an error occurs during the function "Read all Data".
Delete data records from devices not ready for communication	If this option is enabled, the data record for a device is deleted if there is no communication with the device during execution of the function "Read all Data".

PLC programs cannot be read out in the current version of NORD CON. Because of this, the programs of the device and the project file are compared during the action "Read all Data". If they are not identical, a warning is given in NORD CON. This action is skipped if no PLC program is saved for a device.

If device parameters are saved in the project file, this is indicated with a special device symbol in the project structure. The same applies for the PLC program. However, the existence of the device symbol does not provide any information with regard to the current status and completeness of the data. After readout, the parameters can be edited with the parameter editor. The user selects a device in the project tree and opens the parameter editor (Parameterise F7). The parameters can be read or deleted in the editor. The action "Save" saves the parameters for the selected device in the project and also saves the project on the hard drive. If the parameter is to be saved in a separate file, the action "Save as" must be executed.

### Information

If errors occur during "Read all Data", these are noted in the log and the backup is continued. All of the parameters which are noted in the log are not saved in the project file. We recommend that the fault is remedied and a new backup carried out for the device.

### Restore

This function can be executed after opening the project via the main menu. For this, the parameters which are saved in the project file are sent to the devices. As standard, all parameters are always sent to the devices. However, in most cases this is not advisable and only takes up time. To reduce the number of parameters, the user must enable the option "Only transfer enabled parameters" and enable the required parameters in the parameter editor.



In the second step, the PLC programs which are saved in the project are loaded, translated and also sent to the device. As in normal mode, the PLC program for a device is edited with the PLC editor. When the editor is opened, the PLC program is automatically loaded from the project file. After editing, the program can be saved again in the project file with the action "Save". If the PLC program is to be saved in a separate file in this mode, the action "Save as" must be executed.

---

** Information**

If errors occur during the process, this is noted in the log and the process is continued. All of the parameters which are noted in the log could not be saved in the device. The same applies for the PLC programs. We recommend that the fault is remedied and the action restarted.

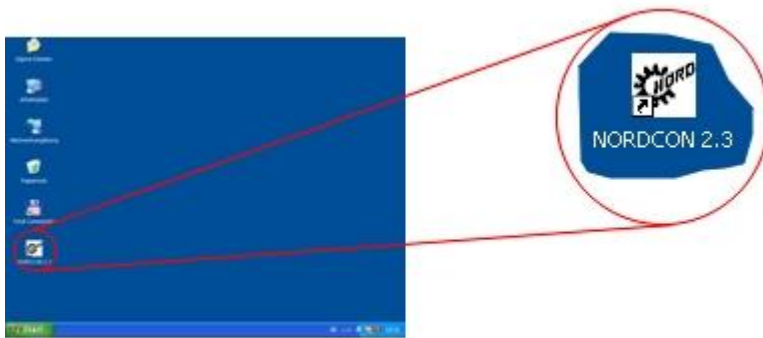
---

## 12 Firmware

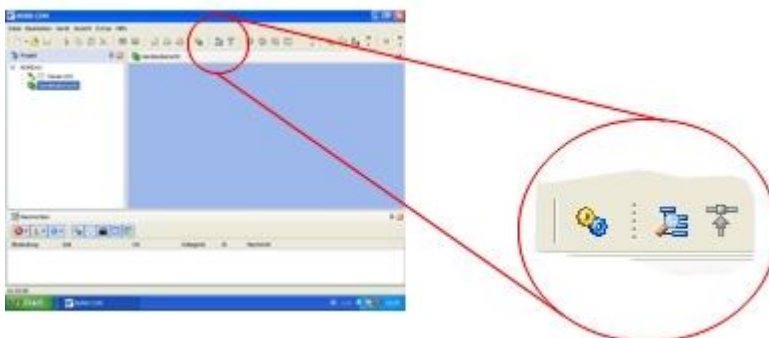
### 12.1 How to update the firmware

The following input steps must be performed:

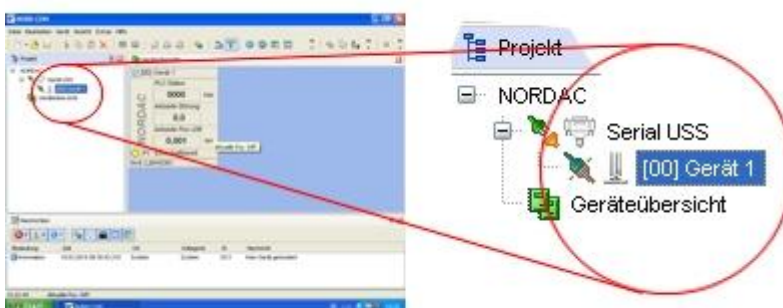
1. Start NORD CON



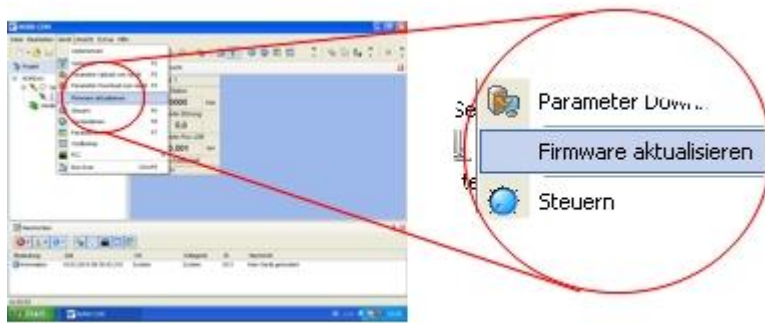
2. Carry out a device search



3. Highlight the required device in the project tree



4. Start the firmware update program via the menu item "Device -> Update firmware"



5. Click on Connect



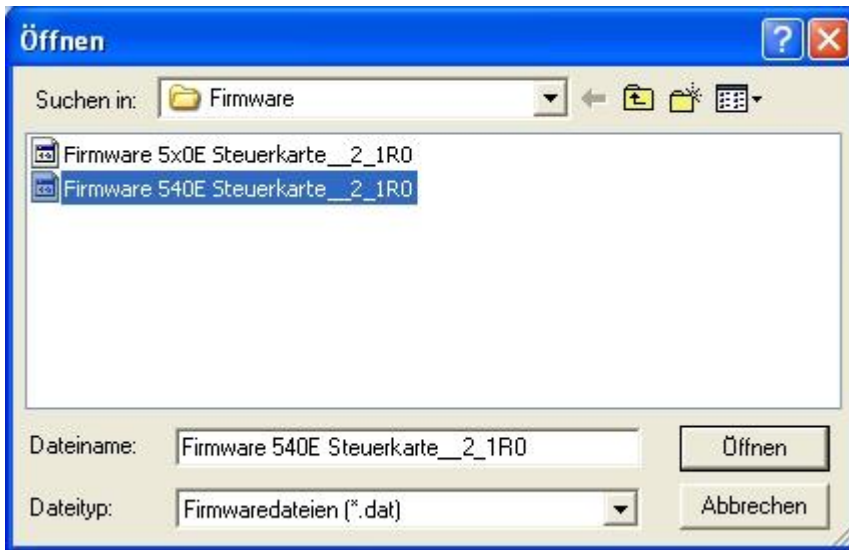
6. Carefully read the warning information and confirm with the "I accept" button



7. Select a firmware file with the aid of the "..." button



8. Select the firmware file and confirm with "Open"



9. Start the firmware transfer with the "Start" button



### **i** Information

The firmware can only be updated if the device has the address 0 and the baud rate is set to 38400 bits/s.

### **i** Information

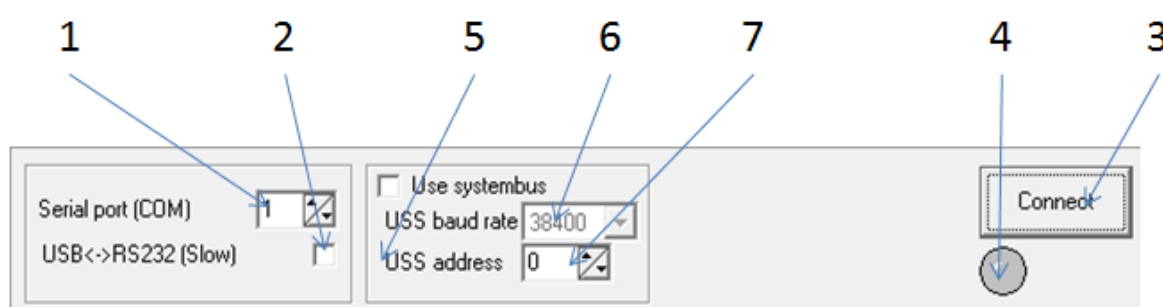
Restart the device if the firmware transfer is interrupted or is not carried out correctly. If the device is not subsequently found in a bus scan, the firmware updated program (FirmwareUpd.exe) can also be started manually. The program is located in the main directory NORD CON.

## 12.2 Firmware update program

### 1. Settings

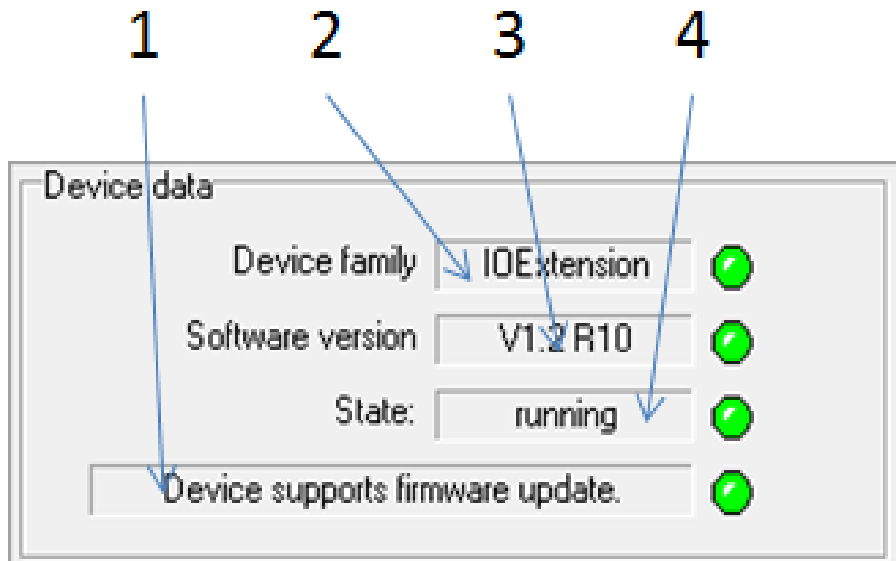
No.	Description
1	In the selection box, the user specifies the COM port of the PC to which the frequency inverter is connected. If the program was called up via NORD CON this parameter does not need to be set.
2	For some USB to RS232 converters, this setting may enable more stable communication. Only select this setting if you have problems with the connection.
3	The "Connect" button creates a connection to the connected device. If a device was found, the LED (4)

No.	Description
	illuminates green and the firmware download window opens.
4	The LED indicates the connection status. Grey - Connection not yet established. Green - The program is connected to a device. Red - No device found.
5	This option activates the update of the firmware via the system bus.
6	This selection box specifies the transfer speed between the PC and the connected device.
7	This selection box specifies the mapped USS address of the device.



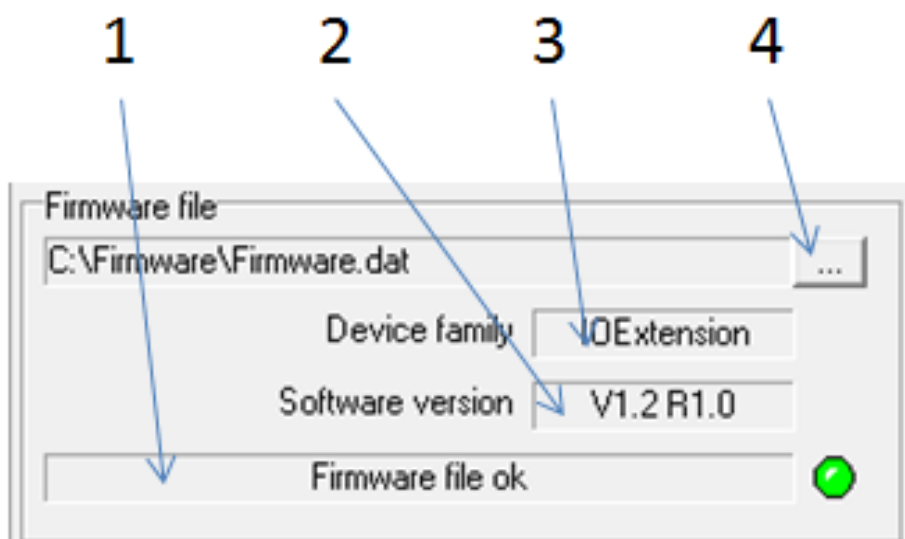
## 2. Frequency inverter data

No.	Description
1	This field indicates whether the frequency inverter which is connected supports firmware updates. If this is not the case, the LED next to the field lights up red.
2	This field displays the frequency inverter family of the connected device. If a device cannot be detected, the LED next to the field is red, and a firmware update is not possible.
3	This field displays the version number of the connected frequency inverter.
4	This field displays the status of the connected frequency inverter. If the frequency inverter is enabled, the LED next to the field is red and a firmware update is not possible.



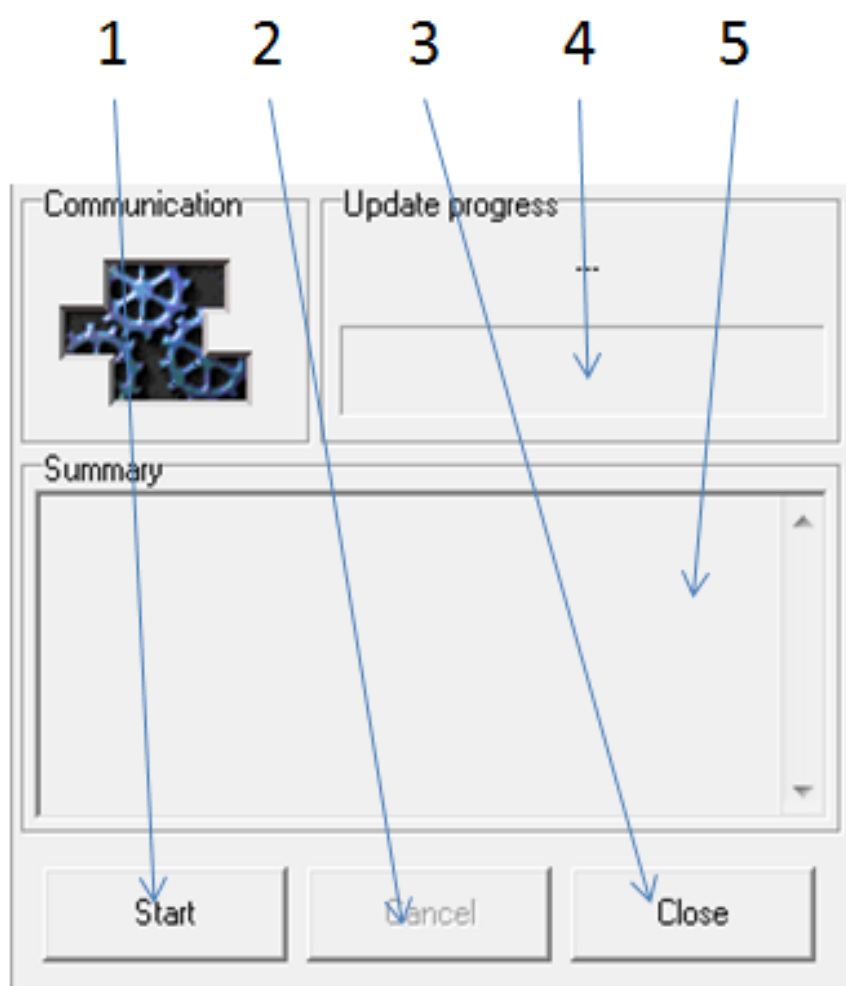
### 3. Select firmware update file

No.	Description
1	This field shows the status of the currently loaded firmware. The LED next to the field is red if the firmware file cannot be loaded or the firmware does not match the connected device. A firmware update is not possible.
2	This field shows the version information for the currently loaded firmware.
3	This field shows the frequency inverter family which is supported by the currently loaded firmware.
4	Clicking the "..." button opens a file selection dialogue. The user can select a firmware file in the window. The file is adopted by clicking "Open" and is then saved in the program configuration file.



### 4. Update firmware

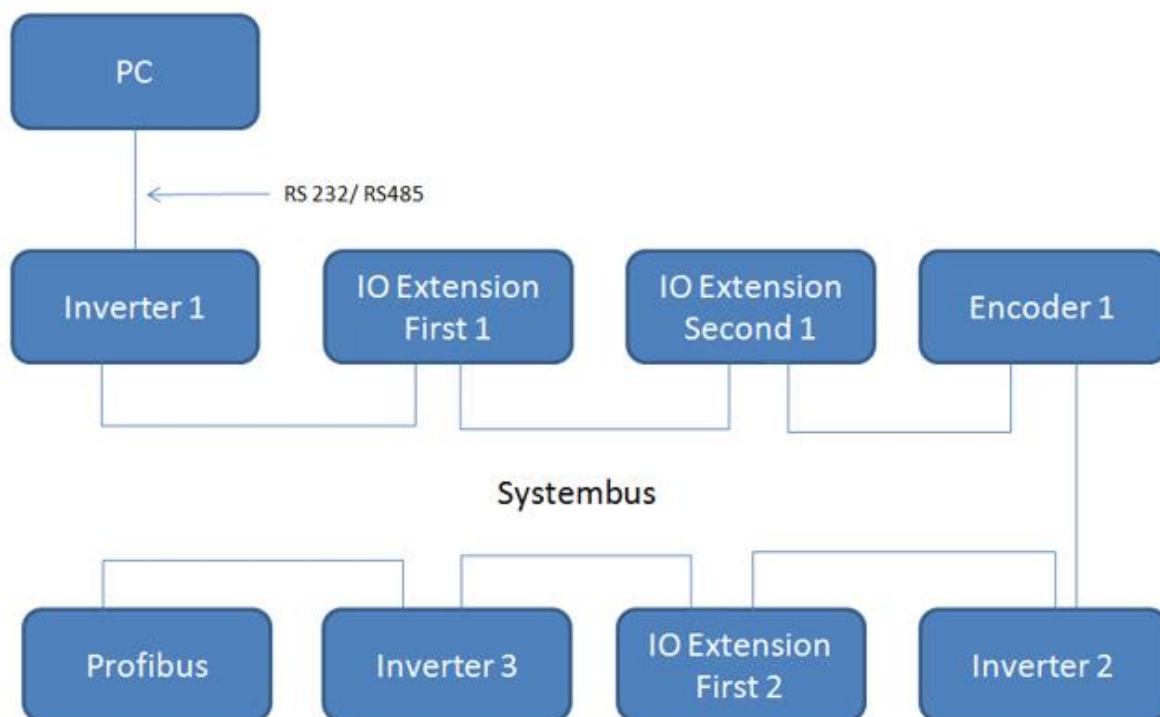
No.	Description
1	The "Start" button must be pressed to start the firmware update. The selected firmware cannot be loaded to the device if the button is not active.
2	An update which has been started is cancelled by pressing the "Cancel" button. Cancellation is only possible during the initialisation phase.
3	The download window cannot be closed during an update. Before or after a download, the user can cancel the update by pressing the "Close" button.
4	The progress display shows the progress of the update and the current status.
5	After the update, the result is displayed in the "Summary" field.



### 12.3 Firmware update via system bus

The system bus is a bus which has been developed by NORD on the basis of a CAN bus. The bus is available for all SK2xxE and SK5xxE frequency inverters with an internal CAN interface, as well as for various additional modules. Up to four frequency inverters, each with 2 additional modules and a CANopen encoder, as well as a bus module can be connected simultaneously, so that in the maximum configuration 17 devices are connected to the system bus. The protocol used for the system

bus corresponds to CANopen. The CAN addresses for the individual inverters have a fixed allocation in the system bus and cannot be changed.



All NORD modules connected to the system bus can be visualised and parameterised via a participant with an RS232/RS485 interface using NORDCON. For this, the communication queries are tunnelled via NORD CON or the device which is connected to the firmware update program. The following mapping procedure is used for tunnelling the queries.

USS address	Module
0	The address must be set in the module which is connected to NORD CON.
1	Frequency inverter 1 (CAN ID: 32)
2	Frequency inverter 2 (CAN ID: 34)
3	Frequency inverter 3 (CAN ID: 36)
4	Frequency inverter 4 (CAN ID: 38)
10	Additional module 1 for frequency inverter 1 (I/O Extension)
11	Additional module 1 for frequency inverter 2 (I/O Extension)
12	Additional module 1 for frequency inverter 3 (I/O Extension)
13	Additional module 1 for frequency inverter 4 (I/O Extension)
19	Frequency inverter with aborted firmware update
20	Additional module 2 for frequency inverter 1 (I/O Extension)
21	Additional module 2 for frequency inverter 2 (I/O Extension)



USS address	Module
22	Additional module 2 for frequency inverter 3 (I/O Extension)
23	Additional module 2 for frequency inverter 4 (I/O Extension)
30	Bus module

The following devices support tunnelling of firmware updates:

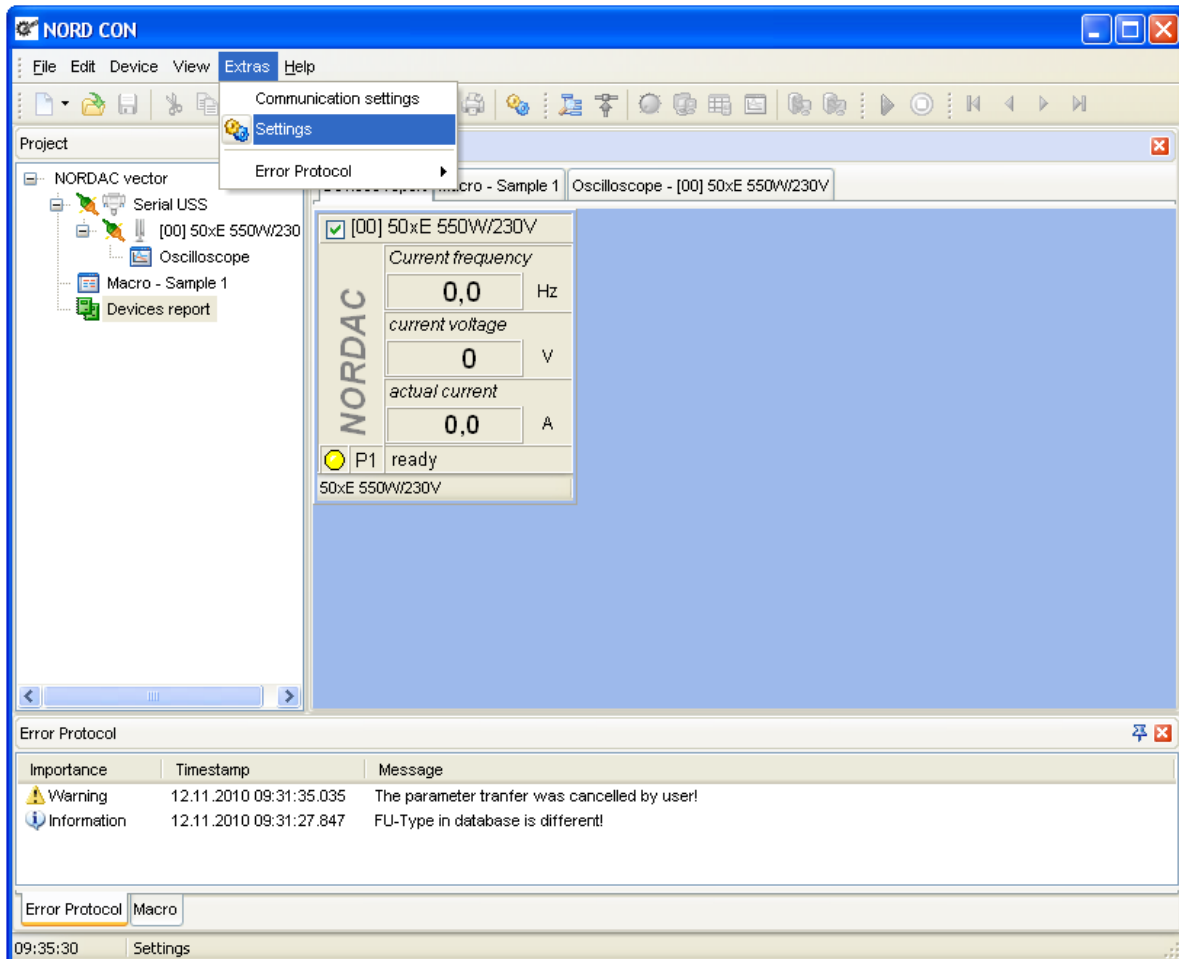
Device	Version
SK 1xxE	All
SK 2xxE	V1.3 and above
SK 5xxE	V2.0 and above
SK 540E	V2.0 and above
SK TU4-DEV	V1.4 and above
SK TU4-CAO	V2.2 and above
SK TU4-PBR	V1.2 and above
SK TU4-POL	All
SK TU4-PNT	All
SK TU4-IOE	V1.2 and above
SK TU4-EIP	All

The following devices support firmware updates via the system bus:

Device	Version
SK 1xxE	All
SK 540E	V2.0 and above
SK TU4-DEV, SK CU4-DEV	V1.4 and above
SK TU4-CAO, SK CU4-CAO	V2.2 and above
SK TU4-PBR, SK CU4-PBR	V1.2 and above
SK TU4-POL, SK CU4-POL	All
SK TU4-PNT, SK CU4-PNT	All
SK TU4-IOE, SK CU4-IOE	V1.2 and above
SK TU4-EIP, SK CU4-EIP	All

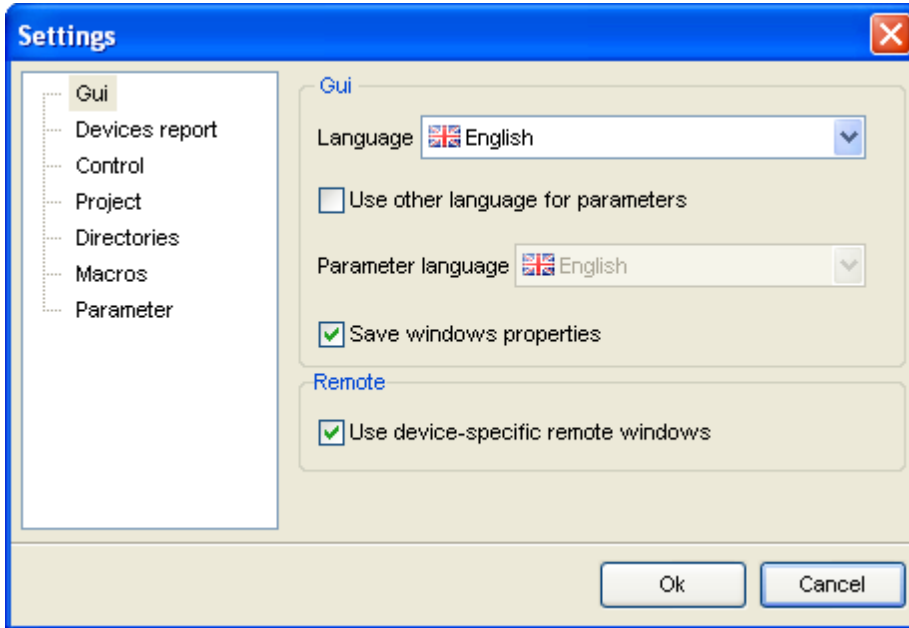
## 13 Settings

The user can change the current program settings with the menu option "Extras->Settings". The attitudes are divided into the following categories:



### 13.1 User interface

In this category the user can change the settings of the user interface. The following options are available:



**Language**

With this option the user can choose the language for the interface.

**Use other language for parameter setting**

With choice of this option the user can choose a different language for the parameter names in the dialog "Parameterisation" in the choice box "Parameter language".

**Parameter language**

With this option the user can choose a different language for the parameter name in the dialog "Parameterisation". This choice is activated by the option "Use other language for parameter setting".

**Save window setting**

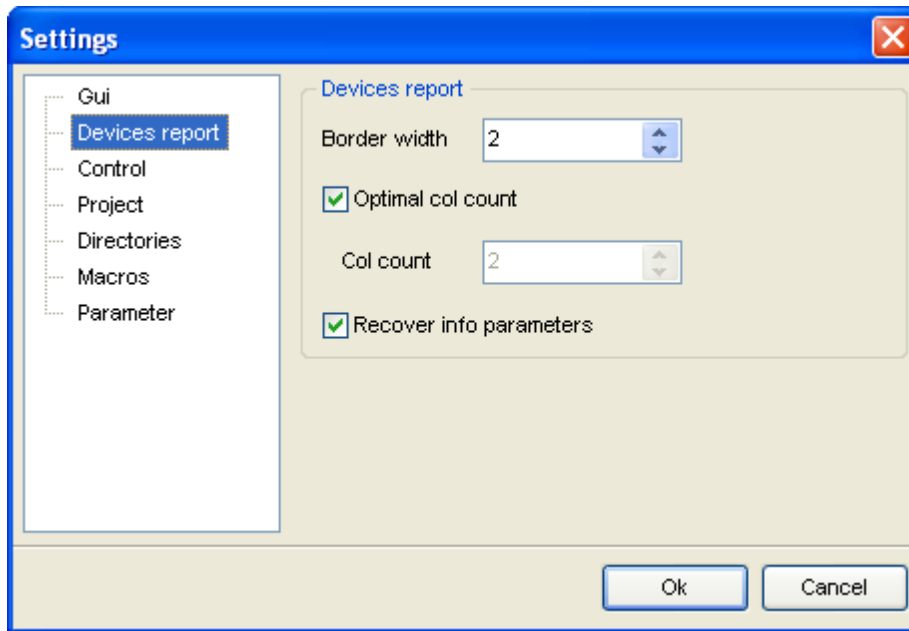
By activation of this option the window settings like position and size is stored and re-activated after opening again.

**Use device-specific remote windows**

If this option is activated, for each type of device special remote windows are produced. Otherwise the standard window is used.

**13.2 Device report**

In the category the user can change the settings of the window "Device overview".



### Border width


With the parameter the user can change the border width of the device display. A value can be set between 0 and 10 pixels. More largely or if smaller value is registered, the largest or smallest value is used automatically.

### Optimal number of columns

If this option is selected then the application calculates the optimal number of columns.

### Number of columns

With this parameter the user specifies a firm number of columns. The value can be set between 1 and 10. If a larger or smaller value is registered, the largest or smallest value used automatically.

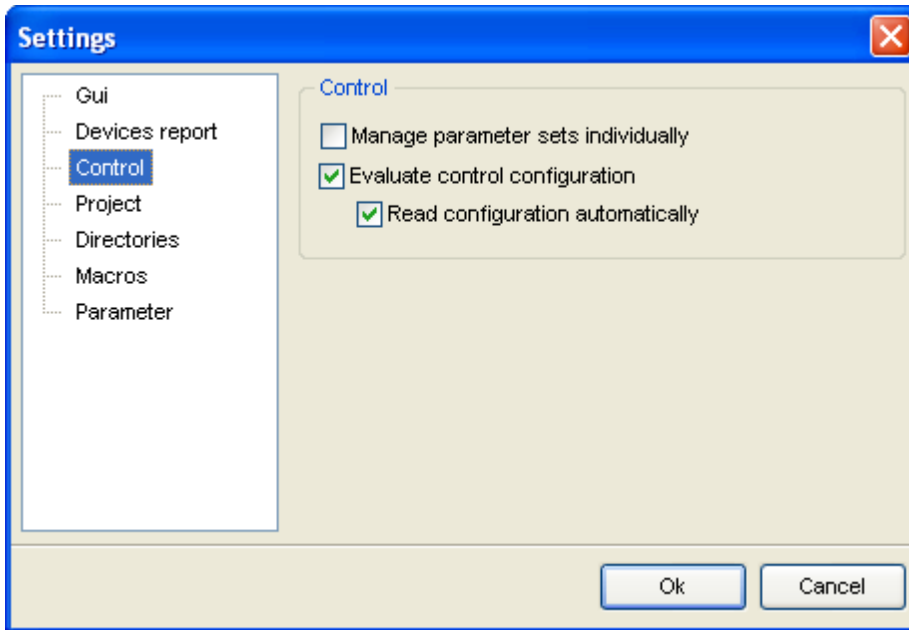
<p>Attention</p> 	<p>This parameter can be only changed, if the option "optimal number of columns" was not selected.</p>
--	--

### Recover info parameters

If this option is selected, the adjusted info parameters of the device are stored and restored with a bus scan or a restart of application.

## 13.3 Control

In the category 5 "Control" the user can change the settings of the "Control" window.



**Manage parameter sets individually**


By activation of the option the setting values and actual values are managed individually in the "Control" window.

**Evaluate control configuration**

The option activated or deactivates the control configuration. With this function being active some functions are released or blocked after checking the configuration. Additionally the names of the parameterised setting value functions or actual value functions are displayed in the window in clear text.

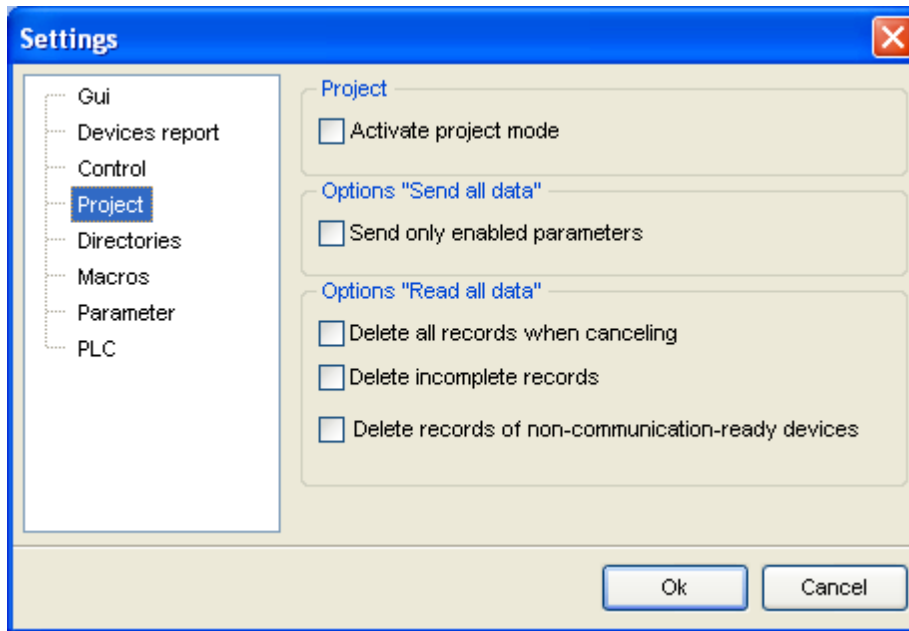
**Read configuration automatically**

The option activates or deactivates the automatic checking of the configuration. With this function activated the control configuration is checked again after focusing of the window.

<p>NOTE</p> 	<p>The function "Evaluate control configuration" is not available in all devices!</p>
---	---

**13.4 Project**

In this heading, the user can specify the path for the project file. Settings such as the interface which is used, bus scan settings, device names, etc. are saved in this file. Old settings can be reloaded by selecting an existing file.



### Activate project mode

The project mode can be enabled or disabled with this option. In project mode, the user can freely parameterise the type and number of devices on the bus. The device parameters and the settings for the application are saved in a project file.

### Only transfer enabled parameters

If this option is enabled, only parameters which have been enabled by the user are sent to the device with the function "Send all Data". As standard, all parameters are always enabled. Enabling of the parameters can be changed in the parameter editor.

### Delete all data records on cancellation

If this option is activated, the data records for all of the devices which are included in the project are deleted when the function "Read all Data" is cancelled.

### Delete incomplete data records

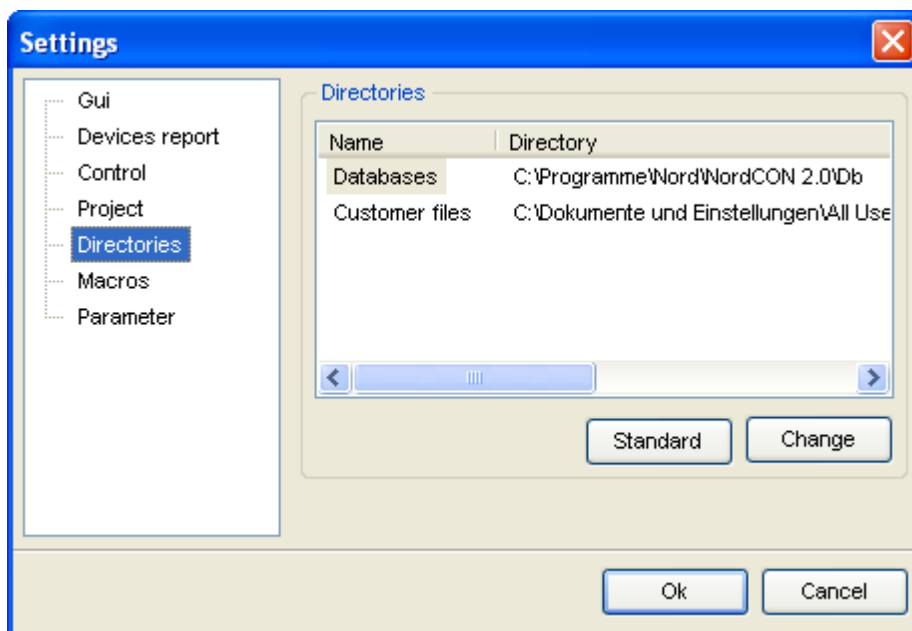
If this option is enabled, the data record for a device is deleted if an error occurs during the function "Read all Data".

### Delete data records from devices not ready for communication

If this option is enabled, the data record for a device is deleted if there is no communication with the device during execution of the function "Read all Data".

## 13.5 Directories

In this heading the directories in which the parameter databases, configuration files, macro files and internal databases are located can be set. In order to change one of the paths, the required directory must be highlighted in the list. A new path can be selected by clicking on the "Change" button. The standard directory for each category can be entered with the aid of the "Standard" button.



### Customer files

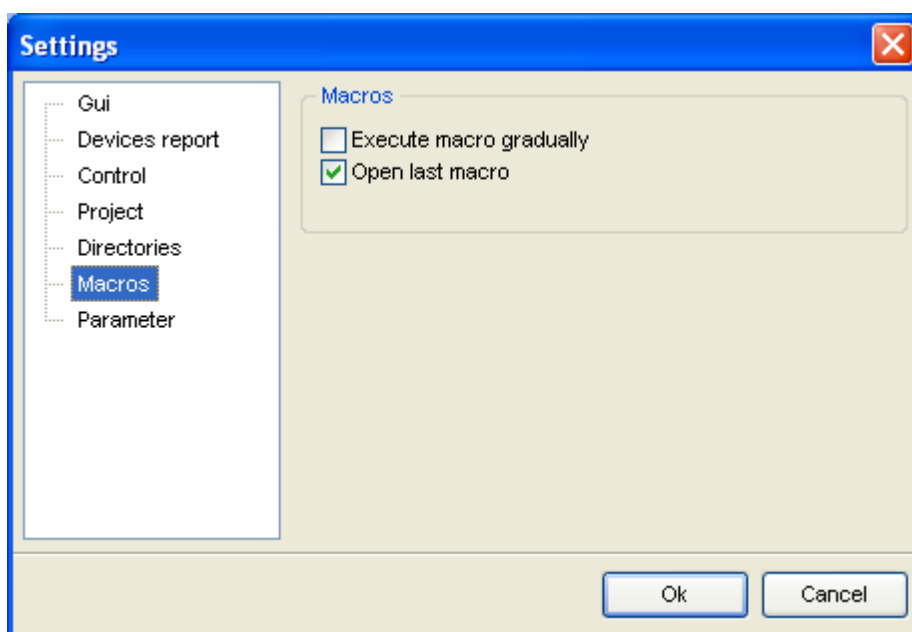
All customer-specific files, e.g. macros or parameter files are saved in this directory.

### Internal databases

These databases are required for the internal execution of the program. The parameter structure for the particular frequency inverter families is saved in these databases.

## 13.6 Macro editor

In the category you can choose settings of 8 "Macro editor" .



### Macro execution step by step

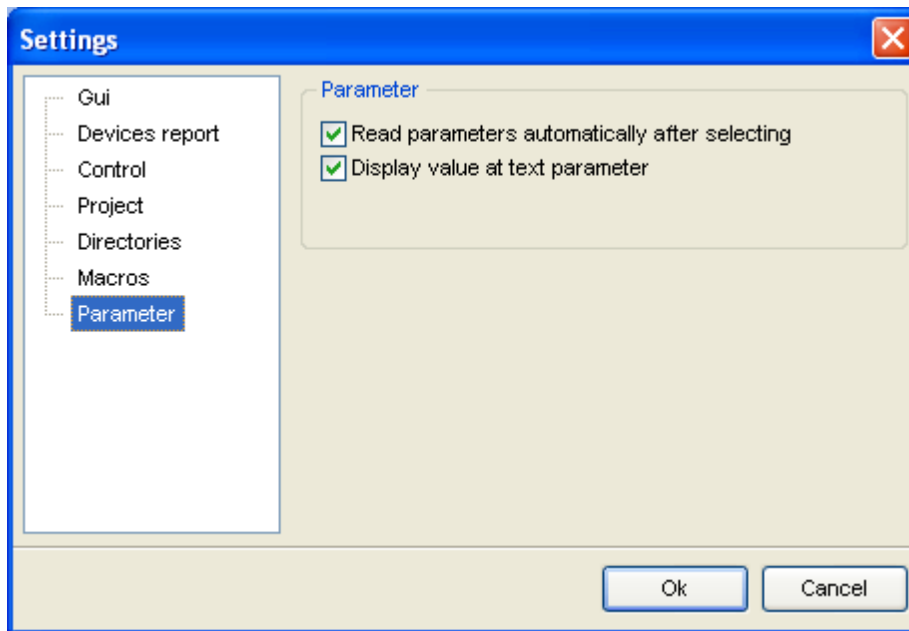
The option activates or deactivates the macro execution step by step. With this option being activated each macro step must be activated separately (cycle/start).

### Open last macro

The option activates or deactivates the function to load the last opened macro.

## 13.7 Parameter

In the category you can choose settings of the 4 "Parameterization".



### Read parameter automatically after selection

The option activates or deactivates the automatic reading of a parameter after selecting.

### Show also the value with text parameter

The option activates or deactivates the display of numerical value with a text parameter.

## 13.8 PLC

### Delete old log entries before compiling

Is this option enabled, the old log entries are deleted before compiling.

### Jump to current breakpoint (debug mode)

Is this option enabled, the line of the current breakpoint is moved into the visual range.



## 14 Messages

### 14.1 Errors and informations

When a fault has occurred, the number of the error by which it is registered in the program is displayed along with a concise error information.

The error messages are to be interpreted as follows:

No	Description
100	Parameter num. Inadmissible
101	Parameter Value cannot be changed
102	Parameter limit exceeded
103	Error in sub-index
104	Not an array
105	Description cannot be changed
106	Description data does not exist
107	Time out receive data
108	Time out send data
109	Error in receive data
110	Different order and answer
200	Could not open serial port!
201	Could not close serial port!
202	First close old serial port!
203	Serial port is not open!
204	The settings of the communication module could not be set. Examine whether the current baud rate are supported.
205	Buffer setup impossible!
206	Timeout setup impossible!
207	Communication not possible!
208	Internal object error!
210	Error writing file!
211	Telegram not created!
212	No high-resolution Timer found!
213	No device found!
214	Only with 16 Bit SetPoint!

No	Description
215	Inverter is running. Close?
216	The firmware update can only be executed if the device address is zero!
217	The firmware update tool could not be started! Please install NORD CON again, in order to repair the problem.
218	Please add a communication module!
219	Do you want to import the file into online view?
220	Here no device can be added!
221	Were found more than 1 device at the bus! An update could cause problems. Would you like to continue?
222	Inconsistency of the control data, if you use macros and control windows simultaneously! Please close all control windows or the macro editor.
223	The transfer cannot be started, because the parameter editor is opened! Please close the editor and restart the function.
224	The on-line help could not be found! Please install NORD CON again, in order to repair the problem.
225	The device cannot be disconnected, because still at least one window of the device is opened.
226	The file cannot be opened. The format of the file is unknown.
227	The file could not be read!
228	The format of the file is unknown!
229	The file was changed by the user
230	The action cannot be executed, because the device is not connected!
231	The settings were changed. Would you like to store the changes?
232	The computer does not support Chinese characters, therefore representation error can occur!
233	The value cannot be converted into INT16!
234	The current version of the device does not support a firmware update over the system bus!
235	The current version of the technology box does not support a firmware update over the system bus!
236	The device at address 0 doesn't support a firmware update over the system bus!
237	The PLC is not registered! Please contact the support (+49 (0)180 500 61 84).
238	The registration code is not correct! Please contact the support (+49 (0)180 500 61 84).
239	The firmware down load can be executed only with a baud rate of 38400 baud!
240	The report cannot be printed, since no printer is installed!
241	The file could not be found on your system!
242	The current version of the technology box TU3 does not support a firmware update!
243	No more devices can be added!
244	The project changed! Would you like to save the project?
245	Failed to contact the device!

No	Description
246	No PLC program could be found for device!
247	Parameter could not be found for the device
248	Transfer cancelled by user!
249	At least one error occurred during the transfer!
250	At least one warning occurred during the transfer!
251	The IP address you have entered is not valid!
252	No more devices can be added!
253	The file is corrupted or is manipulated!
254	In mode "USS over TCP" the firmware update is not possible!
255	The changes require a bus scan! Do you want to accept the modification?
256	The project file could not be found!
257	Please insert a bus module first.
258	It cannot be transferred all settings to the selected bus module! Would you like to continue?
259	An error has occurred during the write operation!
260	PLC program for device is not correct!
261	Do you want to save the project now?
262	IP address is already in use!
263	The directory could not be found!
300	The path for the internal database is not correct!
301	The path for the internal database is not correct. NORD CON will abort.
302	Error while open database!
303	FU-Type in database is not compatible!
304	FU-Type in database is different!
305	Save actual database?
306	Cannot open database!
307	Unallowed folder!
308	Cannot store database!
309	Read all parameter immediately?
310	Please update NORD CON! Faultless parameter transfer isn't guaranteed.
311	Printer isn't correctly installed!
312	Sorry, two parameter windows cannot run simultaneously. View the open window?
313	You have to close to parameter window to quit NORD CON!
314	You have to close to parameter window to make a Busscan!

No	Description
315	A comparison of parameters can be saved only as PDF.
316	The parameters were still not saved permanently in the device. Close anyway?
400	The file could not be loaded, since the file version is unknown!
401	The file could not be loaded, since the file format is unknown!
402	The file was changed by the user!
403	Error open file!
405	No macro file!
406	Empty macro list!
407	Macro list executed!
408	Jump-Label not found!
409	The function cannot be executed, because the scheduler was started.
410	Would you like to save the changes in the macro?
411	The file was changed by the user! Would you like to open the file?
500	Load only the settings?
501	The types of device are different? Would you like to open the file?
502	The file could not be opened, because the version of the file format is unknown!
503	The file could not be opened, because the file format is unknown!
504	The file was changed by the user! Would you like to open the file?
600	Device control is reduced or not possible from the following reason: the control word (P509) is not for USS configures!
601	Device control is reduced or not possible from the following reason: the source of setpoint 1 (P510.0) is not configured for USS!
602	Device control is reduced or not possible from the following reason: the source of setpoint 2 (P510.1) is not configured for USS!
603	Device control is reduced or not possible from the following reasons: the control word (P509) and the source of setpoint 1 (P510.0) are not configured for USS!
604	The controlling of the device is reduced or not possible from the following reasons: the control word (P509) and the source of setpoint 2 (P510.1) are not for USS configures!
605	Device control is reduced or not possible from the following reasons: the source of setpoint 1 (P510.0) and 2 (P510.1) are not configured for USS
606	Device control is reduced or not possible from the following reasons: the control word (P509), the source of setpoint 1 (P510.0) and 2 (P510.1) are not configured for USS!
607	Telegram time-out (P513) is not active!
700	The action cannot be executed, because the connection to the device is interrupted!
701	The remote operation is limited because of the following reason! The parameters are read-only.
800	The parameter transfer was successfully executed

No	Description
801	Errors occurred during the parameter transfer!
802	The parameter transfer was cancelled by user!
803	Errors occurred during the parameter transfer! Would you like to save?
804	The parameter transfer was cancelled by user! Would you like to save?
805	Do you want to see the report?
806	The generation of the report was cancelled by the user!
807	The connection to the devices is now rebuilt! Would you like to continue?
900	Maximally 5 variables can be registered into the watch list!
901	The file must be stored, before you can translate it. Would you like to create a new file?
902	The file could not be opened, because the file format is unknown!
903	The file could not be read!
904	The file was changed by the user! Would you like to open the file?
905	The function is not implemented yet!
906	The PLC program must be saved before you start programming!
907	The PLC program has been changed! Do you want to save?
908	The settings have changed! Do you want to save?
909	PLC format 1.0 not supported.

## 15 NORD DRIVESYSTEMS

### **NORD on the road to success**

NORD was founded in 1965 and now has net sales of approximately 460 million Euro. Our successful climb to the elite list of gearmotor manufacturers is due to our strategy to listen to and work closely with our customers. Together with the help of our customers we have created optimal drive solutions and have had solid growth as a company as well.



### **Global Knowledge and Local Support**

NORD gear is represented in over 60 countries in the world. With more than 3,100 employees to ensure minimum short lead times and fast customer service, you can expect to receive your drive and have your questions answered regardless of your geographic location.

### **Putting Everything in Motion**

With our powerful drive solutions, we put even the “Goliaths” of this world into motion: huge cranes in harbor facilities, retractable roofs sports stadiums, luggage conveyor belts in airports and ski lifts. No matter what your application is, NORD is sure to put it into motion

### **Motoring Ahead**


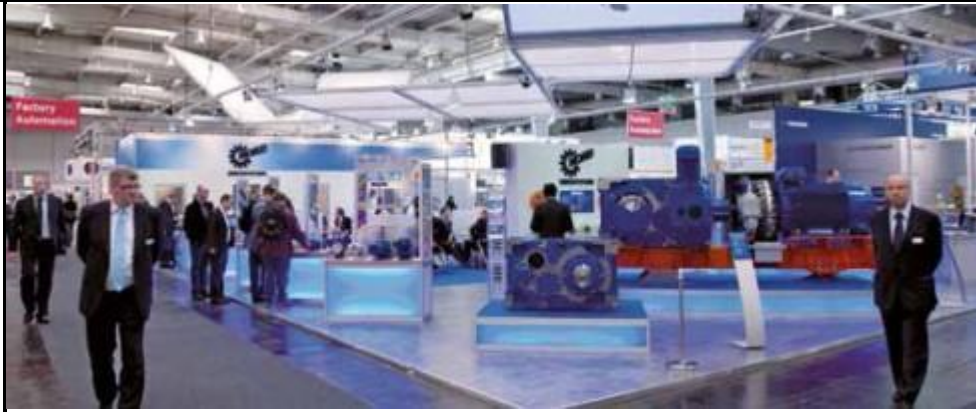
Our products embody an innovative combination between compact mechanics and intelligent electronics. We market and produce a complete product line of mechanical and electronic drive components including, quality gear reducers, motors, frequency inverters, servo controllers and decentralized drive technology.

### **Moving Together**



Our high quality and service standards result from our customer focus. We develop precise fitting, innovative drive solutions based on customer input. Together, NORD and our customers are building long term successful business relationship.

## 15.1 NORD DRIVESYSTEMS corporate history

Ever since NORD was founded in 1965, all companies of the group have adhered to the common strategy of satisfying the demands of our customers.

	
1977	Construction of a modern gear production factory
1979	Establishment of worldwide subsidiaries and the expansion of assembly centres
1980	NORD's UNICASE™ "leak-proof" housing design introduced
1983	Construction of NORD's first motor manufacturing facility
1985	Construction of NORD's first frequency inverter manufacturing facility
1992	Machining factory for castings & steel components built.
	
1997	Construction of motor manufacturing plant in Italy
1998	New assembly plant in France
2000	New assembly plants in Great Britain and Austria
2001	New assembly plant in China
2002	Additional new assembly plant in China
2003	Construction of an assembly plant in Russia
2004	Construction of a new motor plant in Italy
2005	40 years of Nord Gear Opening of the high-rack storage system in Bargteheide, Germany



	Construction of a new assembly centre in China.
2006	New production plant for electronic products opened in Aurich, Germany
2007	Construction of new assembly plants in India and Czech Republic
2008	Expansion Getriebebau NORD, Bargteheide - Construction of a parking garage
	
2009	Expansion at Getriebebau NORD, Bargteheide - Construction of a next high rack storage - Construction of an assembly centre for industrial gear units
2011	NORD DRIVESYSTEMS celebrates the inauguration of the fourth construction stage of the production facility in Gadebusch and the 25th anniversary of NORD GEAR Ltd in Brampton, Canada. In China, NORD celebrates the opening of a second factory in Tianjin, about 100 km south-east of Beijing, while on the fifth continent, the NORD Australian subsidiary is opening in Darwin.
2012	At present NORD DRIVESYSTEMS is represented by 35 subsidiaries throughout the world. The NORD sales and service network is supplemented by sales and service partners in more than 60 countries. With a highly motivated team of employees and a complete range of technologically excellent and high quality drive technology products, the company is ideally equipped to face the challenges of the future.
2013	New construction of a further production facility in Suzhou
2014	Modernisation of the service area and the painting plant at the headquarters in Bargteheide
	
2015	50th company anniversary Construction of a new office building



## 15.2 Frequency Inverters

### 15.2.1 SK 135E



**Performance: 0.25 .... 7.5 kW**

#### **SK 135E – Decentralised motor starter**

Many applications, including those in the field of material handling require electronic starting and stopping of the drive units. NORD has developed the new, innovative motor starter SK 135E for this. Due to its versatility, not only motor starting functions, but also gentle starting or reversing mode are possible. Extensive monitoring functions e.g. protect against overheating. Thanks to the I<sup>2</sup>t triggering characteristic, a motor protection switch is not required. The integrated mains filter of the motor starter SK 135E (with motor-mounting) meets the very highest EMC requirements.

#### **Features and Characteristics**

- Gentle start function
- Reversing function
- Motor or wall-mounting
- IP55, (optional IP66)
- Power range: 3~ 200 240V from 0.25 kW to 4.0 kW 3~ 380, 500V from 0.25 kW to 7.5 kW
- Control and connection of an electromechanical brake
- Integrated mains filter (EMC Class C1 / C2)
- 2 digital inputs
- 2 digital outputs

- Temperature sensor input (TF+/TF-)
- RS232 interface via RJ12 plug
- Optional ATEX Zone 22 3D (in preparation)

Please find further details for the motor starter SK 135E here

### 15.2.2 SK 180E



**Performance: 0.25 .... 2.2 kW**

#### **SK 180E - economical decentralised frequency inverter**

The SK 180E is the answer for all applications in the lower power range, where the main task is speed control. Tried-and-tested NORD know-how is used, so that the proven sensorless current vector control ensures an optimum voltage/frequency ratio at all times. The SK 180E achieves significant advantages for EMC classification. Because of this, a motor-mounted frequency inverter with an integrated mains filter can even be used in a residential environment (Class C1).

### 15.2.3 SK 200E



**NEW - The SK 200E for distributed control versatility 0.33 - 30 hp**

After many years of experience with motor-mounted AC Vector drives, with the release of the SK 200E NORD has now introduced a new series of devices which enable a wide range of decentralised drive technology solutions. These robust, reliable and economic systems are suitable for extensive use in plant applications such as conveying and are specially optimised for price sensitive market segments. Similar to the SK 500E panel mount series, an application-oriented performance level is available which offers the same high quality functionality. Expected features of decentralised components such as robust design, integration of plug connectors, rapid replacement and decentralised modules for communication and I/O signals ensure reliable integration of distributed drive units at the field level.

**Scope of supply - SK 200E:**

- 1~ 115 V 0.25 – 0.75 kW  
1~ 230 V 0.25 – 1.1 kW  
1~ 230 V 0.25 – 4.0 kW  
3~ 480 V 0.55 – 7.5 kW
- Wall-mounted version
- Decentralized modules (also with gateway functionality)

**IP55 protection class as standard. Optional:**

- Size 1 - 3: IP66 (components with "C" = coated)
- Size 4: Component with "C" = coated, with retention of protection class IP55
- Size 1 - 3: ATEX Zone 22, 3D or harsh ambient conditions

**SK 205E Basic Unit**

- High quality control process through sensorless current vector control (ISD)
- External 24DCV control card supply
- 4 control inputs, which can be parameterised to various digital functions
- Externally visible status LEDs (signal state of control inputs) externally adjustable setpoint potentiometers
- Plug-in memory storage module (EEPROM)
- Automatic motor parameter identification
- Four parameter sets, switchable online
- Incremental encoder evaluation (HTL)
- Regenerative, 4 quadrant generator 4Q operation possible by means of optional braking resistor
- PID controller and process controller
- RS 232 & RS485 (RJ12 connector) diagnostic interface

Motor potentiometer function

**SK 215E**

- Basic equipment – as SK 205E (see above)
- Safety function "Safe stop" as per EN 954-1 (EN13849) up to max. Category 4, Stop category 0 and 1.

**SK 225E**

- Basic equipment – as SK 205E (see above)
- ASi interface on board

**SK 235E**

- Basic equipment – as SK 205E (see above)
- Safety function “Safe stop” as per EN 954-1 (EN13849) up to max. Category 4, Stop category 0 and 1.
- ASi interface on board

**15.2.4 SK 500E**



Performance:	0.25 .... 2.2 kW 1/3 AC 200 ... 240 V 3.0 .... 18.5 kW 3 AC 200 ... 240 V 0.55 .... 90 kW 3 AC 380 ... 480 V 0.25 .... 160kW
Output frequency:	0 ... 400 Hz
Manuals	SK 5xxE SK 54xE

## Key word index

<b>A</b>		Firmware update via system bus .....	190
About NORD CON.....	9	First steps .....	9
<b>B</b>		<b>H</b>	
Back up parameters .....	41	Help.....	18
Bus error .....	32	HMI.....	181, 182
Bus scan .....	20	How to update the firmware.....	185
<b>C</b>		How to use NORD CON .....	9
Cancel.....	21	<b>I</b>	
Close.....	13	Import.....	13
Communication.....	193	informations .....	200
Comparison report.....	40	Inserting instructions.....	67
Connect .....	15, 20, 78	Interface and views .....	12
Control .....	9, 15, 44, 45, 78, 193, 195	Introduction .....	9
Copy .....	14	<b>L</b>	
Copying instructions .....	67	Language .....	51, 193
Creating new instructions .....	68	load .....	62
Cut .....	14	Log .....	17
Cutting instructions .....	67	Log window .....	66
<b>D</b>		<b>M</b>	
Delete .....	14	Macro .....	9, 63, 198
Detailed control.....	45, 46	Macro editor .....	193, 198
Device.....	15, 78	Macro generator.....	9, 63, 198
Device.....	20	Main menu .....	12, 13, 15, 78
Device report .....	193, 194	Master (USS Anfrage).....	70
Directories.....	193, 197	Measurement .....	61
Docking.....	26	Measurement .....	59
Down.....	14	Menu .....	12, 13, 14, 17, 18
<b>E</b>		<b>N</b>	
Edit.....	14	Name.....	32
Errors .....	200	New .....	13
export.....	62	NORD CON .....	9
Export .....	13	NORD corporate history.....	206
Extras.....	18	<b>O</b>	
<b>F</b>		Open .....	13
File .....	13	Oszilloskop.....	58, 59, 61, 62
Filter .....	39	Oszilloskop display .....	58
Firmware update program .....	187	Oszilloskop Drucken .....	62

<b>P</b>	
Parameter .....	39, 40, 193, 199
Parameter comparison .....	40
Parameter download .....	42
Parameter download to device .....	15, 78
Parameter language .....	51, 193
Parameter Off-line .....	39
Parameter setup .....	15, 78
Parameter upload from device .....	15, 41, 78
Paste .....	14
PLC .....	74, 199
ABS .....	133
ACOS .....	138
ADD .....	134
ADD( .....	134
AND .....	141
AND( .....	141
ANDN .....	141
ANDN( .....	141
Arithmetical operators .....	133
ASIN .....	138
Assignment operator .....	171
ATAN .....	138
Bit operators .....	141
Bit-wise access to variables .....	171
BOOL_TO_BYTE .....	176
Bus setpoints and actual values .....	160
BYTE_TO_BOOL .....	177
BYTE_TO_INT .....	177
Call-up of function blocks in ST .....	172
CANopen communication .....	78
CASE .....	173
Comments .....	169
Comparisation operators .....	148
configuration .....	83
ControlBox .....	77
ControlBox and ParameterBox .....	162
COS .....	138
CTD .....	108
CTU .....	109
CTUD .....	110
Data processing via accumulator .....	76
Data types .....	168
Data types in ST .....	171
Debugging .....	82
DINT_TO_INT .....	178
DIV .....	134
DIV( .....	134
Editor .....	78
Electronic gear unit with flying saw .....	91
Electronic gear with Flying Saw .....	77
EQ .....	148
Error messages .....	179
Errors .....	166
Evaluation of expressions .....	172
Exit .....	175
EXP .....	139
Extended mathematical operators .....	138
F_TRIG .....	112
FB_FunctionCurve .....	128
FB_PIDT1 .....	129
FB_ResetPostion .....	131
FB_Capture .....	125
FB_DinCounter .....	127
FB_DINTToPBOX .....	120
FB_FlyingSaw .....	92
FB_Gearing .....	93
FB_NMT .....	84
FB_PDConfig .....	85
FB_PDORceive .....	87
FB_PDOSend .....	89
FB_ReadTrace .....	117
FB_STRINGToPBOX .....	123
FB_Weigh .....	132
FB_WriteTrace .....	118
FOR loop .....	174
Function blocks .....	83
Function call-ups .....	170
GE .....	149
GT .....	149
Holding points .....	82
IF .....	173

Image of the process .....	75	MIN .....	135
Info parameters .....	163	MOD .....	136
Input window .....	80	MOD( .....	136
Inputs and outputs .....	151	Motion Control Lib .....	77
Instruction list (AWL / IL) .....	168	MUL .....	136
INT_TO_BYTE .....	178	MUL( .....	136
INT_TO_DINT .....	178	MUX .....	137
JMP .....	175	NE .....	151
JMPC .....	176	NOT .....	142
JMPCN .....	176	Observation points .....	82
Jump marks .....	170	Operators .....	133
Jumps .....	175	OR .....	142
Languages .....	168	OR( .....	142
LD .....	147	ORN .....	143
LDN .....	147	ORN( .....	143
LE .....	150	Overview visualisation .....	120
LIMIT .....	135	ParameterBox .....	77
Literal .....	168	Parameters .....	166
LN .....	139	Process controller .....	77
Loading and storage operators .....	147	Processing values .....	151
loading, saving and printing .....	78	Program Task .....	76
LOG .....	140	R .....	145
LT .....	150	R_TRIG .....	112
MAX .....	135	REPEAT loop .....	174
MC_MoveAbsolute .....	99	Return .....	173
MC_WriteParameter_16 .....	107	ROL .....	144
MC_WriteParameter_32 .....	107	ROR .....	144
MC_Control .....	95	RS Flip Flop .....	113
MC_Control_MS .....	96	S .....	145
MC_Home .....	98	Scope of functions .....	77
MC_MoveAdditive .....	100	Setpoint processing .....	76
MC_MoveRelative .....	101	Setpoints and actual values .....	157
MC_MoveVelocity .....	101	SHL .....	145
MC_Power .....	103	SHR .....	145
MC_ReadActualPos .....	104	SIN .....	138
MC_ReadParameter .....	104	Single Step .....	82
MC_ReadStatus .....	105	Specification .....	74
MC_Reset .....	106	SQRT .....	140
MC_Stop .....	107	SR Flip Flop .....	113
Memory .....	75	ST .....	148
Messages window .....	81	Standard function blocks .....	108

STN .....	148	Restore parameters .....	42
Structured text (ST) .....	171	<b>S</b>	
SUB .....	137	Save .....	13
SUB( .....	137	Save and restore .....	181, 182
TAN .....	138	Save as .....	13
TOF .....	114	Select all .....	14
TON .....	115	Settings .....	18, 51, 193, 196
TP .....	116	Simulate hardware .....	32
Transfer PLC program to device .....	81	Standard .....	19
Type conversion .....	176	Start .....	21
Variables and FB declaration .....	79	Start sequence .....	69
Visualisation .....	77	Stop .....	21
Visualisation with ParameterBox .....	119	<b>T</b>	
Watch- and Breakpoint display window .....	80	Telegram error .....	32
WHILE loop .....	175	Toolbar .....	17, 19, 20, 21
XOR .....	146	Toolbars .....	11, 12, 19
XOR( .....	146	<b>U</b>	
XORN .....	146	Undo .....	14
XORN( .....	146	Undocking .....	26
Port .....	32	Up .....	14
print .....	62	USS Anfrage .....	70, 72
Print .....	13	USS Antwort .....	71, 72
Project .....	11, 12, 17, 18, 21, 193, 196	<b>V</b>	
Project file .....	196	View .....	12, 17, 18, 25
Properties window .....	63	Viewing .....	40
<b>R</b>		<b>W</b>	
Remote .....	12, 18, 25	Window variables .....	63
Remote control .....	15, 78		
Rename .....	15, 78		





**NORD DRIVESYSTEMS Group**

**Headquarters and Technology Center**  
in Bargteheide close to Hamburg, Germany

**Innovative drive solutions**  
for more than 100 branches of industries

**Mechanical products**  
Parallel shaft-, helical gear-, bevel gear- and worm gear units

**Electrical products**  
IE2/IE3/IE4-Motors

**Electronic products**  
Centralized and decentralized frequency inverters  
and motor starters

**7 state-of-the-art production plants**  
for all drive components

**Subsidiaries in 36 countries on 5 continents**  
providing local stock, assembly, production,  
technical support and customer service.

**More than 3,200 employees around the world**  
providing application-specific solutions for our customers.

**[www.nord.com/locator](http://www.nord.com/locator)**

**Headquarters:**

**Getriebebau NORD GmbH & Co. KG**

Getriebebau-Nord-Straße 1

22941 Bargteheide, Germany

Fon +49 (0) 4532 / 289-0

Fax +49 (0) 4532 / 289-2253

info@nord.com, www.nord.com

**Member of the NORD DRIVESYSTEMS Group**

